Guía 8 - Ejercicios integradores: División de responsabilidades
Objetos 1 – UNQ
23/09/2016

Ejercicio 1: Cuentas Bancarias
Modificar el ejercicio Cuentas Bancarias (también conocido como casa de pepe) de la gúia 3.

Parte 1: *Cuentas básicas como clases*
En ese ejercicio hay cuentro objetos cuentas: una normal, una embargada, una dolarizada y una combinada, Convertir ese modelo a uno basado en clases, de manera que pueda haber más de una cuenta normal, más de una embargada, más de una dolarizada y disntintas combinaciones de las mismas.

Verificar si se puede utilizar herencia para no repetir código (¿Se puede dejar en un solo lugar la definición de la variable saldo y las operaciones más básicas para disminuir y aumentar el mismo?)

Parte 2: *Lanzamiento de errores en cuentas básicas*
Resolver lo siguiente para las cuentas normales, dolarizadas y embargadas [1]

Modificar el código para que ante las siguientes situaciones excepcionales, el objeto correspondiente lance un error. **¡Ojo con esto!** No duplicar código

- Cuando una cuenta quiere realizar una extracción o un depósito de un número negativo
- Cuando una cuenta quiere quiere realizar una extracción que dejaría el saldo en negativo

Parte 3: *Cuenta combinada*
Convertir la cuenta com

Parte 4: *Lanzamiento de error en la cuenta combinada*

Analizar el código de la cuenta combinada. Suponer el siguiente escenario: Una cuenta combinada cuya primaria es una cuenta común con saldo 1000 y secundaria una embargada con saldo 500.

- Es necesario escribir algún código adicional para que se lance un error al intentar extraer un número negativo?. Escribirlo si es necesario.

[1] La cuenta combinada es para la siguiente parte

- Es necesario escribir algún código adicional para que se lance un error al intentar extraer un número mayor al saldo, por ejemplo 2000?
- ¿Qué sucede si le envío el mensaje extraer -100 pesos? ¿Queda modificado el estado de la cuenta primaria y/o secundaria?
- Qué sucede si quiero extraer 2000 pesos? ¿Queda modificado el estado de la cuenta primaria y/o secundaria?

Es necesario que cuando el objeto lance una excepción el estado del mismo no haya cambiado, ya que el sistema se podría recuperar del error y el objeto debe estar listo para seguir respondiendo los mensajes de una manera correcta. Por lo tanto el siguiente test debería ser correcto:

```
test "Cuenta Combinada no se modifica ante un error" {

  var cuentaPepe = new Cuenta(1000)
  var cuentaJulian = new CuentaEmbargada(500)
  var combinada = new CuentaCombinada(cuentaPepe, cuentaJulian)

  assert.equals(1500, combinada.saldo())
  assert.equals(1000, cuentaPepe.saldo())
  assert.equals(500, cuentaJulian.saldo())

  assert.throwsException({combinada.extraer(-100)})
  assert.throwsException({combinada.extraer(2000)})

  assert.equals(1500, combinada.saldo())
  assert.equals(1000, cuentaPepe.saldo())
  assert.equals(500, cuentaJulian.saldo())
}
```

Modificar el código para que la ejecución de dicho test sea satisfactoria.

Tip 1: Hacer que todas las cuentas entiendan el mensaje restaurarSaldo", que modifican su saldo por un saldo anterior. El valor del saldo anterior es modificado en cada operación.
Tip 2: Resolver primero asumiendo que las cuentas primaria y secundaria nunca pueden ser otra combinada. Luego adaptar el código para que sí soporte ese caso.

Ejercicio 2: Plagas

Para el estudio de los efectos de las plagas sobre lugares y animales atacados se pide modelar la situación con un sistema modelo con objetos.

Parte 1: *Elementos buenos*

Los lugares o animales, son **elementos** que pueden ser buenos o no para la vida de los humanos. En este sistema vamos a contemplar:

- **Hogar**: Un hogar es bueno si su nivel de mugre es 2 veces inferior al confort que ofrece. Tanto el nivel de mugre como de confort pueden variar de hogar a hogar

- **Huerta**: Una huerta es buena si su nivel de producción supera un valor que es desconocido a priori, pero que es el mismo para todas las huertas.
- **Mascota**: Las mascotas son buenas si su nivel de salud supera a 15

. **Requerimiento:** Saber si es bueno un lugar o un animal.

Parte 2: *Plagas*
Una plaga puede atacar a un elemento (lugar o animal) produciéndole un daño. Este sistema modela cuatro posibles plagas: Cucarachas, mosquitos, pulgas y garrapatas. De cada plaga se conoce la cantidad de bichos que la integra (población). Podría llegar a suceder que transmitan enfermedades. El daño que puede provocar sobre un lugar va a depender de esas cosas.

- Plaga de **cucarachas**: El nivel de daño que produce es la mitad de su población. Transmiten enfermedades si la población es superior a 500
- Plaga de **pulgas**: El nivel de daño que produce es el doble de la población. Transmiten enfermedades siempre.
- Plaga de **garrapatas**: El nivel de daño y la transmisión de enfermedades tienen exactamente el mismo comportamiento que las pulgas.
- Plaga de **mosquitos**: El nivel de daño es el mismo que la población. Pueden transmitir enfermedades o no según cada plaga. Por ejemplo, una plaga podría estar infectada con el dengue y entonces lo transmite.

Requerimientos:

- **Saber el nivel de daño que produce una plaga**
- **Saber si una plaga transmite o no enfermedades**

Parte 3: *Daño sobre las cosas o lugares*
Cuando un elemento es atacado por una plaga, los efectos sobre el mismo dependen del daño con el cual es atacado y si transmite enfermedades o no.

- **Hogar**: La mugre aumenta en la misma cantidad que el nivel de daño recibido independientemente de si la plaga transmite o no enfermedades
- **Huerta**: La huerta baja su nivel de producción en la cantidad de daño recibido. Además podría recibir una disminución extra de 10 si la plaga transmite enfermedades.
- **Mascota**: En caso de recibir una plaga que transmite enfermedades, el nivel de salud disminuye en la cantidad de daño recibido. Si la plaga no transmite enfermedades, entonces no le ocurre nada a la mascota.

Requerimiento: Que un lugar o animal sea atacado por una plaga y produzca sus efectos

Parte 4: *Efecto sobre la plaga al atacar un lugar o animal*
Cuando una plaga ataca a un lugar, además de aplicar los efectos resueltos en el punto anterior, su población crece un 10 %. Salvo en el caso de las garrapatas, cuya población aumenta un 20 %. **Requerimiento:** Que una plaga ataque un lugar o animal, aplicándose los efectos resueltos en el punto anterior y modificando su población.

Ejercicio 3: Seductores

Un grupo de amigos programadores, cansados de fracasar en sus conquistas amorosas, deciden armar, usando la programación orientada a objetos, un sistema que modele esta actividad humana, con el fin de aprender sobre la misma y cambiar su suerte.

Se sabe que el mundo se divide entre los seductores y los seducibles. Un seductor intentará conseguir una cita con un seducible, y en el mejor de los casos, conformará una pareja.

Parte 1: *Aceptar una cita*
. Un seducible nunca aceptará una cita si ya tiene pareja (salvo que el seductor sea su pareja actual).

Tampoco aceptará una cita de un seductor que no sea de un género de su preferencia. Para eso el seductor tiene definido un género (Masculino, Femenino, Transexual, etc...) Y los seducibles definen un conjunto con todos los géneros de su preferencia (ya que podría ser 1 o más).

Además, es posible que un seducible se fije en algunas características del seductor:

- **Cazafortunas:** Solo aceptan una salida con un seductor millonario. Los millonarios tienen un nivel económico mayor a un valor que a priori no es conocido, ya que se ajusta por inflación. Este valor es el mismo entre todos los seductores.
- **Militante:** Necesita que el nivel intelectual del seductor sea superior a un valor que varía entre cada persona. Además, el seductor debe ser capaz de aportar a la causa, esto puede darse por una ayuda económica (el seductor es millonario), ó, porque el seductor tiene un aspecto personal que supera a la mitad de su nivel intelectual.
- **Envidioso:** Un envidioso solo acepta una cita si su rival también la aceptaría
- **Libre:** No le exige nada adicional al seductor

Con respecto a los seductores, todos cuentan con un nivel intelectual, económico y de aspecto personal propio a cada uno (al cual llamaremos base). Pero estos valores base pueden ser afectados positivamente o negativamente por ciertos artilugios que puede poseer el seductor o no.

Nota: Por el momento el sistema acepta 4 tipos de artilugios distintos, pero se espera incorporar más de una manera sencilla, sin necesidad de modificar el código escrito en la clase que modele a los seductores.

Nota: Todos los porcentajes que se utilizan a continuación corresponden ser aplicados sobre el valor base del aspecto en cuestión.

- **Billetera:** Una billetera aumenta el nivel económico en la cantidad de pesos que contiene la billetera. Además, aumenta en 10 % el aspecto personal. Sin embargo, si el seductor es millonario, reduce su nivel intelectual en 20 %.
- **Auto:** Mantener un auto es caro, por lo tanto el nivel económico disminuye en una cantidad que corresponde al 10 % del precio del auto. Sin embargo, el aspecto personal se incrementa en dos veces el precio del auto. No tiene influencia alguna sobre el nivel intelectual.
- **Reputación:** Un seductor que tiene una reputación tiene valores absolutos variables para cada nivel (intelectual, económico y aspecto físico) que pueden ser positivos o negativos. Por ejemplo, un seductor cuyos valores base son: 5 de intelecto, 10 de aspecto personal, y 15 de nivel económico, con una reputación de 10 de para el intelecto, 4 para el aspecto personal y -6 para el económico queda con los valores: 15, 14 y 9 respectivamente.
- **Título de técnico programador:** Aumenta el nivel intelectual en un 100 % y el nivel económico en un 80 % y no cambia el aspecto personal.

Por otro lado, se sabe que estas reglas son levemente distintas para los **seductores estrellas** (*Chuck Norris, Guillermo Francella, Javier Mascherano, entre otro)*, ya que las modificaciones que recibe de sus artilugios son el triple de lo que afectan a los demás seductores.

Ejemplo: el nivel base económico de un seductor es 100, tiene una billetera con 10 pesos y un título de técnico programador. El nivel económico total es 100 + 10 + 100*1.8 = 290. Pero si se tratara de chuck norris, sería 100 + 3 * (10 + 100*1.8) = 670

Requerimiento: Saber si un seducible acepta una cita de un seductor.

Parte 2: *Seducir*

.

Un seductor intentará seducir a un seducible. Si el seducible está interesado, es decir, acepta tener una cita (resuelto en el punto 1) la cita ocurre. Esto puede traer consecuencias (cambios en el estado interno) sobre los artilugios del seductor (según quien es el seducible). Si al terminar la cita, el seducible aceptaría tener una nueva cita, entonces se ha formado una pareja.

Nota: Por cuestiones de simpleza, el seducible sólo puede tener una pareja, pero no hay requerimientos acerca de que el seductor tenga que ser monógamo. Por lo tanto, una

solución que solo acepte una pareja para el seductor es igual de válida que una que acepte múltiples parejas.

Con respecto a las consecuencias de la cita:

- Si fue con un **Cazafortunas**, los artilugios del seductor sufren una consecuencia de índole **económica**.
- Si fue con un **Envidioso**, los artilugios del seductor sufren una consecuencia de índole **económica** y otra de índole **moral**.
- Si fue con un **Libre**, los artilugios del seductor sufren una consecuencia de índole *moral*.
- Si fue con un **Militante**, no ocurre nada.

Es importante saber que:

- La **billetera** que sufre una consecuencia *económica* reduce su dinero en 10 % y no hay cambios en las consecuencias de índole *moral*.
- El precio del **auto** se reduce un 5 % tanto para una consecuencia de índole *económico* como para uno de índole *moral*.
- El **título de técnico programador** es inmune a las consecuencias *económicas* y *morales*
- La **reputación** disminuye 10 en el nivel de economía en caso de una consecuencia *económica*. En el caso de una consecuencia *moral*, disminuye 5 en cada uno de los los niveles intelectual y aspecto físico.

Requerimiento: Que un seductor seduzca a un seducible, obteniendo las modificaciones en los estados de los objetos intervinientes en cada caso:

- Actualización del estado interno de cada uno de los artilugios del seductor en caso de ocurrir una cita.
- Actualización de la pareja del seducible si corresponde.

Parte 3: *Colecciones*

Resolver cada uno de los siguientes requerimientos:

1. Un seductor debe ser capaz de identificar y devolver sus **artilugios económicos capos**: Son aquellos que consiguen mejorar el nivel del seductor en un valor importante. Para saber si un artilugio es capo, existe la siguiente fórmula:

   modificación que aporta económicamente al seductor ¿ 0.5 * nivel básico de economía del seductor

2. **Ostentación**: Cuando un seductor ostenta, devuelve un atributo económico capo cualquiera.

3. Dada una instancia de la clase **Cupido** que tiene una colección de seducibles, obtener todos los **seductores flechados**. Considerando como *flechado* el hecho de que sean pareja de un seducible de la colección de ese objeto cupido.

4. **Elementos ostentosos de los flechados**: Es la colección que se forma con todos los elementos con los cuales ostentan los seductores flechados por cupido.

Ejercicio 4: Mascotas

Un grupo de fanáticos de las mascotas quiere un sistema que permita registrar cómo afectan las mismas al estilo de vida de sus dueños y cuales son los mejores entrenadores de mascotas.

Parte 1: *Felicidad del dueño*

Se sabe que los dueños pueden tener muchas mascotas y que influyen en los mismos en muchos niveles, lo cual determina si el dueño es o no feliz. Los niveles a tener en cuenta son el amor, la diversión y la seguridad.

Un dueño se considera feliz si se siente seguro y además, su nivel de amor sumado a su nivel de diversión es mayor o igual al doble de su nivel de seguridad. Un dueño se siente seguro cuando el nivel de seguridad es mayor a 50.

Para calcular los distintos niveles que influyen en la felicidad, hay que tener en cuenta:

- **Amor:** *"uno cosecha lo que siembra"*: Un dueño tiene un valor propio de amor que reparte entre sus mascotas en partes iguales. **Su nivel de amor es la sumatoria del amor que le devuelven sus mascotas**. Tener en cuenta que el valor recibido por cada mascota es calculado (amorPropioARepartir / cantidadDeMascotas).
- El nivel de **seguridad** es el mínimo entre la edad del dueño y 50; a lo cual se le suma todos los valores de seguridad que le aportan sus mascotas.
- El nivel de **diversión** depende de una base que es distinta para cada dueño. A eso se le suma todos los valores de diversión que le aportan las mascotas.

Por el momento, el sistema ofrece las siguientes alternativas de mascotas:

- **Canarios**
  - Un canario no aporta nada a la seguridad de su dueño.
  - Tampoco le devuelve amor.
  - Con respecto a la diversión, aporta siempre 5 unidades con su bello canto.
- **Gatos**

- Los gatos no son los mejores animales para aportar a la seguridad de su dueño, pero está comprobado que algunos de ellos son territoriales, por lo tanto el valor de seguridad puede variar de individuo a individuo.
- Con respecto al amor, devuelve un porcentaje adicional al amor recibido. Ese porcentaje no se sabe a priori cuál es. Pero si se sabe que **es el mismo para todos los gatos**.
- La diversión que aporta varía entre cada animal.

- **Perros**
  - Los perros son los mejores guardianes, su nivel de seguridad es de **30 más su nivel de guardián más un extra** de 20 en el caso que el dueño sea menor de 18 años.
  - Con respecto al amor, los perros devuelven el doble del amor que reciben de su dueño. Con excepción de los *labradores*, que suman 5 puntos de amor extra a lo que aportaría un perro de otra raza.
  - Los perros divierten mucho más a los pequeños que a los grandes, por eso se calcula como **125 menos la edad del dueño**. Con excepción a los *labradores*, que siempre aportan **130**.

Requerimiento: Saber si un dueño es feliz. **Nota:** Considere un caso en que un dueño tiene una mascota de cada tipo. **Nota:** para calcular el mínimo de dos números puede usar el mensaje `min()`. Ejemplo: `1.min(2)` devuelve 1

Parte 2: *Entrenamiento*
Las mascotas pueden mejorar algunos de los niveles que aportan al recibir entrenamiento. Todas las mascotas **tienen un entrenador**. La eficacia del entrenamiento depende de los **puntos de entrenamiento** que puede aportar cada entrenador. El entrenador de la mascota puede ser su mismo dueño o un entrenador contratado.

- Los puntos de entrenamiento que aporta un dueño son los mismos puntos de amor que le otorga a cada mascota (Según el punto anterior).
- Un entrenador contratado aporta como puntos de entrenamiento su experiencia, que varía entre cada entrenador.

La manera en que afecta el entrenamiento a cada mascota es:

- Sin importar cuantos puntos de entrenamiento se aporte a un canario, éste no aprende nada nuevo.
- Cuando un perro es entrenado, su nivel de guardián aumenta en un 10 % de los puntos de entrenamiento que otorga el entrenador.
- Cuando un gato es entrenado, el nivel de diversión que aporta aumenta en un 15

Requerimiento: Decirle a una mascota *entrenar* y que la misma sea afectada según las reglas anteriores (las cuales dependen del tipo de mascota y de los puntos de entrenamiento que pueda aportar su entrenador). Escribir un pequeño test o líneas de REPL en el cual quede claro cómo se establece quien es el *entrenador* de una mascota.

Parte 3: *Mejor entrenador*
. Un dueño quiere saber cuáles de sus mascotas son **buenas**. Una mascota es buena si la suma de los niveles de amor, seguridad y diversión que le aportan a su dueño superan los 150 puntos.

Los entrenadores de las mascotas buenas son todos **entrenadores recomendables** por el dueño.

El **mejor entrenador** según un dueño, es aquel **entrenador recomendable** que aporta mayor cantidad de puntos de entrenamiento a las mascotas.

Nota: Las colecciones entienden en le mensaje `max(unBloqueTransformer)` para calcular el máximo. Por ejemplo: `['hola', 'Leonardo', 'como', 'te', 'va'].max(x => x.size())` devuelve 'Leonardo' **Requerimiento:**

- Determinar las mejores mascotas de un dueño
- Determinar los entrenadores recomendables por un dueño
- Determinar el mejor entrenador según un dueño.

Nota: No repetir código!

Ejercicio 5: Emisoras de radio

Un grupo multimedio que maneja varias emisoras de radio está organizando la programación para el próximo año, renovándola totalmente para todas las emisoras. La programación de una emisora se compone de un conjunto de programas, que van "de tal hora a tal hora" (p.ej. de 10 a 13). Para simplificar, suponemos que todos los días va la misma programación, y que todos los programas empiezan y terminan "en punto". O sea, no se contempla la posibilidad de que un programa p.ej. termine 13.30, o termina a las 13.00 o termina a las 14.00. Todos los programas de estas emisoras son musicales.

Algunos programas son organizados por la emisora, a estos se los llama *programas autónomos*. Para cada programa autónomo se define qué estilos musicales incluirá (rock, pop, electrónica, clásica, etc.). P.ej. un programa autónomo puede definir que incluirá pop y electrónica

Los otros programas son armados a la medida de una empresa, que paga para que un determinado programa se llame p.ej. "la hora Acme"; a estos se los llama *programas financiados*. Las características de un programa financiado se ajustan a los deseos de la empresa que lo financia. Una misma empresa puede financiar más de un programa, en la misma o distintas emisoras. Para estos programas, los estilos musicales que incluirá son

los de las bandas que prefiere el financiador. De cada banda se conoce a qué estilo musical se dedica.

De los deseos de cada empresa financiadora nos van a interesar: los grupos musicales que prefiere

P.ej. una empresa financiadora puede ser Acme S.A., que prefiere a Ratones Paranoicos, Intoxicados y Babasónicos. Los estilos musicales que incluirán los programas que financie Acme serán rock (Ratones e Intoxicados) y pop (Babasónicos).

Realizar un modelo en objetos de la situación descripta en el que se pueda resolver los siguientes requerimientos:

- Saber si un programa se puede incorporar a la programación de una emisora determinada. Las condiciones son:
  - Que no se superpongan los horarios con otros programas ya incluidos en la programación de la estación, p.ej. si ya hay un programa de 10 a 12, no se puede incorporar otro que va de 11 a 13
  - Que alguno de los estilos musicales incluidos en el programa esté dentro de los estilos que se definen para la emisora.
- Agregar un programa a la emisora. Si el programa no se puede incorporar se debe lanzar un error.
- Saber en qué programas de la programación de la emisora puede incluirse un determinado tema musical. De cada tema se sabe el grupo musical que lo interpreta y la duración. Se debe cumplir con las siguientes condiciones:
  - Para los programas financiados el grupo esté entre los preferidos de la financiadora, mientras que para los programas autónomos, hay que verificar que el estilo musical del grupo esté entre los estilos musicales incluidos en el programa.
  - Además, si el tiempo de los temas ya asignados al programa más el del tema que se quiere averiguar supera el tiempo total del programa, entonces no se puede incorporar el nuevo tema; esto se debe corroborar tanto para los programas autónomos como para los financiados.
- Saber si un programa está cargado, un programa se dice cargado si tiene temas asignados para más del 80 % de su tiempo total.

Ejercicio 6: Oktubrefest

Ein Prosit! En Münich-Alemania, todos los años se festeja la Oktubrefest. Para esta gran feria de comidas, entretenimientos y mucha cerveza nos piden a construir un programa en objetos que modele el comportamiento de las personas en la fiesta. Al entrar en la fiesta se pueden encontrar enormes carpas cerveceras, en donde muchísima gente se reune a... bueno... tomar cerveza. Queremos controlar la entrada a estas carpas dependiendo de la disponibilidad de la carpa y los gustos del público.

Jarras y Carpas
Las carpas cerveceras tienen un límite de gente admitida, algunas tienen una banda para tocar música tradicional, y por supuesto que todas venden jarras de cerveza. De cada jarra de cerveza sabemos su capacidad en litros y de qué marca es la cerveza. Cada carpa sirve jarras de cerveza de sólo una marca (que depende de cada carpa).

Persona
De cada persona se sabe su peso, la jarras de cerveza que compró hasta el momento, si le gusta escuchar música tradicional o no, y su nivel de aguante, que es un número. Una persona está ebria si la cantidad de alcohol en sangre que tiene multiplicado por su peso supera su aguante. Además, de cada persona interesará saber qué marcas de cerveza le gustan. Se sabe que los belgas toman sólo cerveza con más de 4 gramos de lúpulo por litro, a los checos les gustan las cervezas de más de 8 % de graduación (ver abajo qué es la graduación de la cerveza), a los alemanes les gustan todas.

Marcas
Existen varias marcas de cerveza. Están las marcas de cerveza rubia (como la Corona), las marcas de cerveza negra (como la Guiness), y las marcas de cerveza roja (como la Hofbräu). De cada marca se sabe su contenido de lúpuluo, o sea, cuántos gramos de lúpulo por litro llevan.
La graduación de una cerveza es su porcentaje de alcohol en volumen. P.ej. una cerveza de 10 % de graduación, tendrá 0,1 litro de alcohol por litro de cerveza. Se sabe que todas las cervezas rubias tienen cada una una graduación distinta, y que las marcas de cerveza negra tienen una graduación que se calcula como el mínimo entre la graduación reglamentaria y el doble de su contenido de lúpulo. La graduación reglamentaria es mundial, claro está, y puede cambiar con el tiempo. La cerveza roja se hace con métodos similares a la cerveza negra, teniendo un 25 % más de graduación.

Se pide REPL o test de uso , codificación completa, y diagrama de clases para los siguientes **requerimientos**:

1. Saber cuántos litros de alcohol aporta una jarra de cerveza. Ej: una jarra de cerveza de medio litro de la marca 'Hofbräu' (suponiendo que tiene 8 % de graduación alcohólica) aporta 0,5 * 0,08 = 0,04 litros de alcohol.

2. Saber el total de litros de alcohol que ingirió una persona (en base a las jarras de cerveza que compró).

3. Saber si una persona está ebria.

4. Saber si una persona quiere entrar a una carpa, es decir, si la carpa vende una marca de cerveza que a él le guste y si cumple su preferencia sobre que haya o no haya música.

5. Saber si una carpa deja ingresar a una persona, o sea, si dejándola entrar no supera su límite de personas y si la persona no está ebria.

6. Saber si una persona puede entrar a una carpa, es decir, si quiere entrar a la carpa y la carpa lo deja entrar.

7. Que una persona entre a una carpa. Resolver la situación de una persona que quiera ingresar a una carpa y no podía por la falla de alguna condición resuelta en los puntos anteriores.

8. Saber cuantos ebrios empedernidos hay dentro de una carpa. Los ebrios empedernidos son los ebrios que todas las jarras que compraron, son de 1 litro ó más.

Ejercicio 7: La Familia

Se sabe que hay locales en varias zonas de Buenos Aires (por ejemplo: Retiro, Almagro, Palermo Sensible, Parque Avellaneda, etc) controladas por diferentes familias poderosas pertenecientes a la mafia, que manejan sus negocios y se rigen por estrictas normas de honor. El honor lo vamos a manejar por unidades a las que vamos a llamar "puntos de honor", que vamos a describir más adelante. No es ninguna noticia que las familias son muy ambiciosas. Las familias se atacan entre sí para conquistar alguna propiedad ajena, eliminando a su competencia con el objetivo de obtener más poder. A los integrantes de cada familia que pueden participar en los ataques se los considera integrantes criminales, mientras que hay otros integrantes que son miembros respetables de la sociedad (como abogados, funcionarios públicos, etc) que trabajan para enmascarar las actividades delictivas (y obviamente, jamás participan en los ataques). Los integrantes criminales llevan encima sus instrumentos de trabajo a.k.a. armas. Dentro del arsenal que pueden manejar, vamos a considerar

- Cuchillos: otorgan a quien lo posee una cantidad de puntos de honor que se establece para cada cuchillo; tienen una potencia destructiva de 1.
- Ametralladoras: otorgan a quien la posee una cantidad de puntos de honor que es igual para todas las ametralladoras; tienen una potencia destructiva que se establece para cada ametralladora.
- Bombas: no otorgan puntos de honor, y tienen 1000 de potencia destructiva.

Un arma se considera heavy si tiene más de 200 de potencia destructiva, u otorga más de 10 puntos de honor. El honor se maneja así:

- Si es un integrante criminal, es la suma de los puntos de honor que le otorgan sus armas más una cantidad de puntos de honor básicos que son de él.
- Si es un integrante respetable, es la suma de los puntos de honor básicos más 10 por cada título universitario que posee.

Algunos de los integrantes son considerados capos. Para saber si un integrante es considerado capo se tiene en cuenta (otra vez) si es criminal o respetable: Un integrante criminal es capo si tiene más de 100 puntos de honor y tiene al menos un arma heavy. Un integrante respetable es capo si tiene más de 100 puntos de honor y tiene un cargo político (Se sabe de cada integrante respetable si posee un cargo político o no)

Se desea conocer:

1. Si un arma es heavy.
2. Los puntos de honor de un integrante.
3. Si un integrante es capo.
4. El conjunto de integrantes capos de una familia.
5. El honor "per cápita"de una familia, que se calcula como la sumatoria de los puntos de honor de sus integrantes dividido la cantidad de miembros (promedio)
6. Si una familia es ejemplar. Las familias ejemplares son aquellas cuyos miembros tienen más de 60 de honor.

Ejercicio 8: Picadito de Quidditch

El Quidditch es un deporte que juegan los magos y brujas de todo el mundo, donde se enfrentan 2 equipos volando con escobas mágicas en un campo con forma oval que en cada uno de los lados, uno para cada equipo, está dotado de tres aros ubicados a un distinto nivel de altura. Cada equipo está conformado por: un guardián, dos golpeadores, tres cazadores y un buscador. Para jugar se utilizan 3 tipos de pelotas distintas:

- Quaffle: Los cazadores deben meter la quaffle en alguno de los aros del equipo oponente para sumar a su equipo 10 puntos. El guardián debe proteger los aros y evitar que le metan un gol. Hay sólo una en el campo de juego.
- Bludger: Las bludgers son pelotas que aportan emoción (y violencia) a los partidos; los golpeadores pueden batear una bludger apuntando hacia un jugador del otro equipo y los más grosos hasta pueden evitar goles desviando la quaffle.
- Snitch: El buscador es el encargado de buscar la snitch y atraparla; al hacerlo suma 150 puntos para su equipo y termina el partido.

Queremos armar un sistema usando el paradigma orientado a objetos para modelar este extravagante juego. Se pide REPL o tests de uso, diagrama de clases y métodos para:

Parte 1: *Carecterísticas básicas*

a Saber el nivel de manejo de escoba de un jugador. Esto se calcula como los skills del jugador / su peso.

b Saber la velocidad de un jugador, que es la velocidad de la escoba que use multiplicado por su nivel de manejo de escoba. Actualmente existen estas escobas (pero podrían agregarse más):

- Nimbus: La velocidad se calcula como 80 - la cantidad de años desde su fabricación por el porcentaje de salud de la misma. Por ejemplo, una Nimbus 2001 con un 50

- Saeta de Fuego: es la escoba más veloz fabricada, la velocidad es de 100 km/h.

c Saber la habilidad de un jugador. Como cada jugador tiene un entrenamiento acorde a su posición en el campo, su habilidad se calcula de formas diferentes:

- Cazadores: su velocidad + sus skills + su puntería * su fuerza.
- Guardianes: su velocidad + sus skills + su nivel de reflejos + su fuerza.
- Golpeadores: su velocidad + sus skills + su puntería + su fuerza.
- Buscadores: su velocidad + sus skills + su nivel de reflejos * nivel de visión.

Parte 2: *Jugadores*

a Si un jugador **le pasa el trapo** a otro; esto sucede si es por lo menos el doble de habilidoso que el otro jugador.

b Si un jugador **es groso**, que se cumple si su habilidad es mayor al promedio de su equipo y su velocidad mayor a un valor arbitrario que a medida que el mercado de escobas mejora se actualiza para todos por igual.

c Si un equipo **tiene un jugador estrella** para jugar contra otro equipo. Un jugador es estrella si le pasa el trapo a todos los jugadores del equipo contrario.

Parte 3: *Turno de un equipo*
Hacer un equipo juegue un turno contra otro. Cuando esto pasa cada jugador del equipo juega **un único turno** de acuerdo a su puesto, como se describe a continuación...

- El **golpeador** cuando juega un turno elige un **blanco útil** (ver 4.b) del equipo contrario al azar [2]. Si puede golpear a su blanco, el mismo sufre los efectos de **ser golpeado por una bludger** (ver 4.c) y el golpeador sube sus skills en 1. Para poder golpear al otro debe cumplirse que la puntería del golpeador sea mayor que el nivel de reflejos del blanco. No pasa nada si pifia.
- El **guardián** no hace nada en su turno, sólo participa activamente cuando hay que bloquear.
- El **buscador** cuando juega un turno intenta obtener la snitch. Cuando comienza el partido los buscadores arrancan buscando la snitch, si la encuentran deben perseguirla hasta atraparla. **Qué hacer cuando le toca jugar, depende de su estado en ese momento**.
  - Si el buscador está **buscando la snitch**, debe hacer un random entre 1 y 1.000 y si el número obtenido es menor a su habilidad + la cantidad de turnos continuos buscando entonces encontró la snitch.

[2] el mensaje anyOne() de las listas retornan un elemento al azar.

- Si está **persiguiendo la snitch**, debe recorrer 5.000 kms para poder atraparla. En cada turno recorre una cantidad de kms igual a su velocidad / 1,6. Una vez que la atrapó, aumenta sus skills 10 puntos y su equipo gana 150 puntos.

Parte 4: *Más sobre el juego*

a Saber si un jugador **puede bloquear** el tiro de un cazador. Esto se cumple para los *golpeadores* si son grosos, para los *guardianes* si sacan 3 en un random de 1 a 3 [3], y para los *cazadores* si le pasan el trapo al cazador que tiró. Los *buscadores* no bloquean.

b Saber si un jugador es un **útil**. Se cumple para un *cazador* si tienen la quaffle, para un *buscador* si está persiguiendo la snitch y le faltan menos de 1000 kilómetros y para un *guardián* si su equipo no tiene la quaffle. No es útil golpear *golpeadores*.

c Hacer que un jugador **sea golpeado por una bludger**. Cuando esto pasa el jugador pierde 2 puntos de skills y su escoba recibe un golpe (las Nimbus pierden 10 % de salud, a las Saetas de Fuego no les pasa nada). Además los *buscadores* deben **reiniciar la búsqueda** y los **cazadores**, si tenían la quaffle, **la pierden**.

Parte 5: *Más y más sobre el juego*

a Cuando un buscador groso es golpeado por una bludger, el mismo queda aturdido un turno y en el siguiente puede retomar su actividad tal cual estaba. Un buscador aturdido es blanco útil si la actividad a retomar lo llevaría a ser útil en el siguiente turno. **Pista: si el punto 3 está bien hecho no habría que cambiar nada de lo codificado para agregar esta lógica.**

b El **cazador** cuando juega un turno, **lo que hace depende de su posesión de la quaffle** . Si no tiene la quaffle, no hace nada. Si tiene la quaffle, debe intentar meter gol, que implica:

- Evitar bloqueos: cada uno de los jugadores contrarios **intenta bloquear** el tiro del cazador, y en caso de **poder bloquearlo** (ver 4.a) se interrumpe el tiro. Lógicamente en cuanto uno bloquea, los jugadores restantes no deben seguir bloqueando.
- Si nadie lo pudo bloquear, el equipo gana 10 puntos por meter gol, y el cazador gana 5 puntos de skills

Si el tiro se interrumpe por un bloqueo, el cazador pierde 2 puntos de skills y el que bloqueó gana 10. Independientemente de si pudo meter gol o si fue bloqueado, el cazador **pierde la quaffle**. **Siempre** que un cazador pierde la quaffle, ésta es atrapada por el cazador rival más rápido (y el rival pasa a estar en posesión de la quaffle). **Ayuda**: No hace falta modelar las pelotas (como la quaffle), sino lo que se hace cuando la tengo, y lo que se hace cuando no. ¡Delegar bien!

[3] los numeros entienden el mensaje randomUpTo(otroNumero). Por ejemplo 1.randomUpTo(4) devuelve azarozamente 1, 2 o 3

Ejercicio 9: Faceless

Un grupo de inversores, que no quiere dar la cara, nos contrató para llevar a cabo una red social. Faceless permite a los usuarios publicar diferentes tipos de contenidos, los cuales incluyen texto, fotos y videos, entre otros. Por cuestiones de almacenamiento en los servidores, nos interesará saber cuánto espacio ocupa cada publicación, en KB:

- Las **fotos** tienen un alto y ancho dado en pixels; el espacio que ocupan se calcula como (alto * ancho) * factor de compresión. El factor de compresión actual para las fotos es de 0.7, pero probablemente cambiará en el futuro.
- Las publicaciones de **texto** son mucho más fáciles de calcular, ya que el espacio que ocupan es igual a la cantidad de caracteres que tiene.
- Los **videos**, tienen un tamaño que depende de la calidad que tenga: para la calidad normal, el tamaño es igual a la duración del video en segundos. Para los videos HD el tamaño es igual al triple de la duración en segundos del video. Deben poderse agregar en un futuro más calidades sin modificar el código existente.

Los usuarios que pueden ver una publicación pueden indicar que esa publicación les gusta, aumentando el número de "me gusta"s de la misma. A Faceless le importa tanto la cantidad de "me gustas" que recibió una publicación, como saber quienes son los usuario que le dieron "me gusta".

Los usuarios de Faceless tienen "amigos", pero no quieren compartir todas sus publicaciones con todos ellos. Por ejemplo, hay fotos y videos que no quieren que sus familias vean, por alguna extraña razón. Para satisfacer esa necesidad, cada publicación tiene asignado un permiso, que puede ser:

- público: cualquiera puede ver la publicación
- sólo amigos: sólo los amigos pueden verla
- privado con lista blanca: los usuarios que pertenezcan a la lista pueden verla
- privado con lista negra: los usuarios que no pertenezcan a la lista pueden verla.

Se pide:

1. Saber cuánto espacio ocupa el total de las publicaciones de un usuario.
2. Poder darle "me gusta" a una publicación, y conocer cuántas veces fue votada de esta forma.
3. Saber si un usuario es más amistoso que otro: esto ocurre cuando tiene más amigos
4. Saber si un usuario puede ver una publicacion (porque tiene permisos).
5. Determinar los mejores amigos de un usuario. Esto es, según la gente de faceless, el conjunto de sus amigos que pueden ver todas sus publicaciones.

6. Saber cual es el amigo más popular que tiene un usuario. Es decir, el amigo que tiene mas "me gusta" entre todas sus publicaciones.

7. Saber si un usuario *stalkea* a otro. Lo cual ocurre si el stalker le dio "me gusta" a más del 90

Armar los siguientes escenarios como tests o ejecuciones en el REPL:

- Hacer que un usuario publique un contenido dado, con un permiso dado.
- Publicar un video para que esté accesible sólo para los amigos de un usuario.

Ejercicio 10: Regalos Navideños

Hay un grupo de gente interesada en gestionar los regalos navideños y las decoraciones de las casas, para esto nos pidieron un programa con objetos que satisfaga sus necesidades.

Un árbol cualquiera tiene una vejez, una altura y una cantidad de varas. Básicamente en el mundo existen dos grandes filosofías *arbolísticas*, están los árboles **naturales** y los árboles **artificiales**. De los árboles naturales sabemos que la capacidad total (en cantidad) que tienen para contener cosas es equivalente a su vejez multiplicado por su altura. En cambio los árboles artificiales pueden alojar tantas cosas como cantidad de varas tengan más su altura.

A su vez en un árbol navideño (independientemente de su material) pueden haber muchas cosas:

- Regalos, de cada regalo se sabe su precio y destinatario.
- Tarjetas, de cada tarjeta se saben sus destinatarios (por ejemplo, un saludo del primo lejano para sus 3 primitos). El precio de una tarjeta es siempre de $2 y no ocupa lugar en el árbol, por lo que no importa cuantas tarjetas haya, esto no influye en su capacidad disponible.
- Adornos: todo adorno tiene un peso base que es distinto para cada adorno, el precio de un adorno cualquiera se determina como su peso multiplicado por su coeficiente de superioridad (claramente hay adornos que se creen mejores que otros). A su vez hay algunos adornos que requieren un trato especial:
  - Las tiras de luces, su coeficiente de superioridad es un décimo de la luminosidad que aporta en el ambiente (la luminosidad de una luz es equivalente a la cantidad de lamparitas que tiene)
  - Las figuras elaboradas, tienen todas el mismo coeficiente de superioridad.

  Un adorno ocupa lo mismo que un regalo: un lugar del àrbol.

Se pide código completo, workspace de uso, y diagrama de clases para:

1. Saber la capacidad total de un árbol navideño. Dar dos soluciones:
   a) Una usando herencia.
   b) Una usando composición.
2. Saber la capacidad disponible de un árbol.
3. Saber cuántas cosas están destinadas para alguien en un árbol (los adornos están destinados para todos). Por ejemplo, en un árbol que tenga una tarjeta para Juan, un regalo para Pedro, y una figura elaborada, hay 2 cosas para Juan. Tip: delegar. (como siempre!)
4. Agregar una cosa a un árbol. Sólo se puede agregar si entra, y si el destinatario de la misma tiene menos de 2 cosas para él en el árbol. Si no se puede agregar, que tire un error con un mensaje descriptivo.
5. Poder conocer el costo total gastado en un árbol, que contempla todo lo gastado en ese árbol.
6. Agregar al punto 1a ó al punto 1b los árboles frondosos, que son árboles naturales cuya capacidad total es más grande. Se le suma capacidad para una cosa cada dos varas.
7. Tests
   a) Considerando el punto 1b, construir un escenario que cree un árbol natural sólo con una tarjeta.
   b) Hacer un test que verifique que el punto 1 ande bien. (Se puede modificar el escenario anterior).
   c) Hacer un test que verifique que no se agregue un regalo si no hay espacio. (Se puede modificar el escenario anterior).