Paradigmas de Programación
Integración

Ejercicio 1 - Siete y medio

Hay un mazo infinito, una cantidad de jugadores y la banca. La banca reparte inicialmente una carta a cada jugador, y después le da a cada jugador una carta extra hasta que se planta o se pasa de siete y medio. Después juega la banca misma.

Se juega con las cartas de truco: para cada palo (oros / copas / bastos / espadas), las cartas de 1 a 7, la 10 (sota), la 11 (caballo) y la 12 (rey); las sotas, caballos y reyes se conocen como figuras.
Para el juego las cartas de 1 a 7 valen su número, y las figuras valen medio. El palo no tiene ninguna influencia.
Por ejemplo: si tengo un 2, un 4 y un 11, mi puntaje es 6 y medio. Si tengo un 3 y un 5 mi puntaje es 8 y me pasé de siete y medio.
Sí me interesa conocer, por un lado el valor de la carta para el siete y medio, por otro el número (para poder mostrarlo).

Hay que determinar qué jugadores ganaron y cuáles perdieron. Ganan los que tienen más que la banca, y además no se pasaron. P.ej. si la banca hace 6 y medio, ganan los jugadores que hicieron 7 o 7 y medio.

Suponer que cada jugador se planta en una cantidad determinada (p.ej. uno en 6, otro en 5, etc.); idem la banca.

Complicación: introducir un factor de azar en la decisión de parar.

Una forma no mucho más complicada de hacerlos sería con un mazo no infinito sino real y que la decisión de plantarse sea por la probabilidad de pasarse, por ejemplo > a un 50 % de acuerdo a las cartas que quedan en el mazo, la banca sería algo más especuladora ya que tambien tendria en cuenta a cuantos les gana si se planta para definir el limite de riesgo aceptable para pedir otra carta. Es algo parecido pero más realista me parece.

Consejos

Vayan de a poco, p.ej.

- Modelen una carta
- Modelen un jugador que recibe cartas y sabe su puntaje.
- Agréguenle al jugador la capacidad de responder a la pregunta de si se planta o no.
- Modelen un mazo infinito, acá tienen que resolver cómo largar cartas al azar.
- Modelen la mecánica del juego con los jugadores (x ahora sin la banca), que reparta una carta a c/u y después juega con c/u hasta que decide plantarse.
- Agreguen a la banca, que juega al final.

- Agreguen la determinación de ganadores y perdedores, ¿a qué objeto de los que pusieron tiene sentido preguntarle esto?
- Complicaciones: determinación de parada al azar, mazo no infinito y que cada jugador puede analizar.

Ejercicio 2 - Empresa de transporte

Una empresa de transportes tiene distintos tipos de vehículos: camionetas, combis, camioncitos y camionazos.
Cada uno de ellos varía en tiempo para realizar un viaje, consumo de combustible y carga máxima.
La empresa cubre varias rutas, de las cuales conoce: cantidad total de km, y de estos cuántos están en buen estado y cuántos en mal estado.

Para las camionetas tenemos:
- carga máxima: 1200 kg.
- tiempo para realizar un viaje: (km buen estado / 120) + (km mal estado / 100).
- consumo de combustible: 0.1 litros x km.

Para las combis tenemos:
- carga máxima: 2500 kg.
- tiempo para realizar un viaje: km / 100 hasta 1800 kg, km / 90 si es más de 1800 kg.
- consumo de combustible: 0.15 litros x km hasta 1500 kg, 0.2 litros x km si es más de 1500 kg.

Para los camioncitos tenemos:
- carga máxima: 6000 kg.
- tiempo para realizar un viaje: 1/4 de hora fijo + (km buen estado / 100) + (km mal estado / n), donde n es 75 hasta 3000 kg, 70 entre 3000 y 5000 kg, 60 por más de 5000 kg.
- consumo de combustible: 1 litro x km en buen estado x 1500 kg de carga + 1.5 litro x km en mal estado x 1500 kg de carga.

Para los camionazos tenemos:
- carga máxima: 20000 kg.
- tiempo para realizar un viaje: 1 hora fijo + (km buen estado / 90) + (km mal estado / 70), con carga hasta 10000 kg. Más de 10000 kg sumar un 30%.
- consumo de combustible: 1 litro x km x 1000 kg de carga + 0.3 litros x km fijo.

Armar los objetos necesarios para poder conocer
- determinar de entre los vehículos de la empresa, cuáles pueden hacer un viaje. Un viaje se define por ruta, una carga, y un tiempo máximo.
- asignar un viaje a un vehículo. Por ahora no nos importa el momento en que se hace cada viaje (eso viene después).
- determinar el consumo total de los viajes asignados.
- conseguir la ruta entre dos lugares. Los lugares se representan mediante Strings.

Complicación #1 - Mantenimientos

Contemplar mantenimientos programados para los vehículos.
Para cada entrada de mantenimiento se definen día de entrada y día de salida.
Para un mismo vehículo pueden programarse varios mantenimientos a futuro.
Un vehículo no puede asumir viajes mientras está en mantenimiento, y no puede entrar en mantenimiento en un período que se solapa con el de algún viaje asumido, o con el de otro mantenimiento programado.
Agregar la posibilidad de desasignar un viaje asignado, para aumentar la libertar de los operadores de manejar viajes y mantenimientos.

Complicación #2 - Empresas amigas

La empresa agrega la posibilidad de firmar convenios con otras empresas amigas, para tener la posibilidad de pasarle viajes.

Cada empresa amiga declara un conjunto de rutas, que son las que puede asumir. Una empresa puede asumir cualquier viaje sobre las rutas que declara, sin limitación de carga, tiempo de viaje, ni disponibilidad.

Cada empresa amiga tiene una tarifa por cada ruta que ofrece, que se establece así: tarifa base hasta 3000 kg más un adicional por cada 1000 kg o fracción adicionales.
Cuando se le asigna un viaje a una empresa amiga, aumenta la deuda con ella; cada tanto se le realizan pagos.

Hacer los agregados y/o modificaciones para que cuando pido quiénes pueden hacer un determinado viaje, además de los vehículos incluya las empresas amigas que lo pueden hacer.
O sea, una única Collection en la cual tal vez algunos son vehículos y otros son empresas amigas. conocer el saldo que se tiene con cada empresa amiga, resultante de los viajes que se le asignaron y los pagos que se le hicieron.
Obtener un detalle de cuenta corriente de una empresa amiga, donde consten los viajes asignados y los pagos hechos, con día y monto de cada uno. La fecha que debe tomarse para la asignación de viajes es el día en que se asigna, no el día del viaje.

Ejercicio 3 - Eventos de Catering

Una empresa que organiza eventos ofrece una comida como parte de cada evento.
Cada evento tiene un menú único, que incluye entrada y plato principal; algunos menúes incluyen postre mientras que otros no.

La empresa trabaja con varias cocinas, que tienen capacidad de producción limitada de distintos platos. P.ej. una cocina puede tener una capacidad de producción de 50 porciones de milanesas, 25 porciones de ensalada mixta, y 180 porciones de flan.
Siempre la capacidad se mide en porciones por día.

También trabaja con casas de comida rápida (McDonalds y similares), que de los platos que prepara tiene capacidad de producción infinita.

Armar los objetos necesarios para poder determinar

- si una cocina (o casa de comidas rápidas) puede o no preparar un menú completo, o sea si tiene capacidad de producción > 0 para cada uno de sus platos.
- dado un evento (que se define por menú y cantidad de gente) qué puede proveer una cocina (o casa de comidas rápidas), en platos y cantidad de porciones de cada plato, en un día.
- dado un evento, armar alguna forma de conseguir en un día todos los platos necesarios combinando entre un conjunto de cocinas y/o casas de comida rápida.

Consejo: resuelvan los tres requerimientos de a uno, lo que tienen que modelar cada uno los va a ayudar para el siguiente.
Cuando hayan terminado, pregúntense en dónde están usando polimorfismo, y en dónde están usando delegación.

Ejercicio 4 - Manejo de mercadería

Una empresa quiere informatizar parte de su manejo de mercadería, en particular conocer el costo de la mercadería que saca a la venta.
La empresa tiene tres tipos de productos:

- comprados: los compra y los revende así como los compró.
- conservados: los compra, los conserva en condiciones especiales por un tiempo, y después los vende.
- fabricados: los fabrica.

Para organizarse, la empresa arma lotes que consolidan los productos que va a obtener en un determinado período.
Cada lote se compone de un conjunto de órdenes de obtención de productos, para dividir el trabajo entre los distintos grupos de personas que trabajan en la empresa.
Finalmente, cada orden indica que deben obtenerse determinados productos, cada uno en una cantidad que se especifica.
Por ejemplo: una orden puede indicar que deben obtenerse

- 10 hogazas de pan blanco mediano
- 8 hormas de queso sardo
- 6 tarros de mayonesa

Acá los productos son "hogaza de pan blanco mediano", "horma de queso sardo" y "tarro de mayonesa".
El costo de una orden se calcula así:

costo orden = suma (costo producto * cantidad del producto en la orden)

El costo de un producto se calcula así:

costo producto = costo producción + costo almacenaje

El costo de producción depende del tipo de producto, de esta forma:

- productos comprados: **precio de compra**
- productos conservados:

  precio de compra original + días de conservación del producto * peso del producto en kg. * costo de conservación x día x kg.)

El costo de conservación por día y por kg es el mismo para todos los productos conservados.
Por ejemplo, si la horma de queso sardo lleva 45 días de conservación y pesa 8 kg, el costo de conservación x día x kg es de 100 pesos, y el precio de compra original de la horma de queso sardo es 3000 pesos, su costo de producción será

*3000 + (45 * 8 * 100) = 39000 pesos*

Para estandarizar, se toma un peso promedio x producto (en el ejemplo, que todas las hormas de queso sardo pesan 8 kg).

- productos fabricados:

  cantidad de hs de trabajo que lleva producirlo * costo de la h/trabajo

  El costo de la hora/trabajo es el mismo para todos los productos fabricados.

El costo de almacenaje de un producto es:

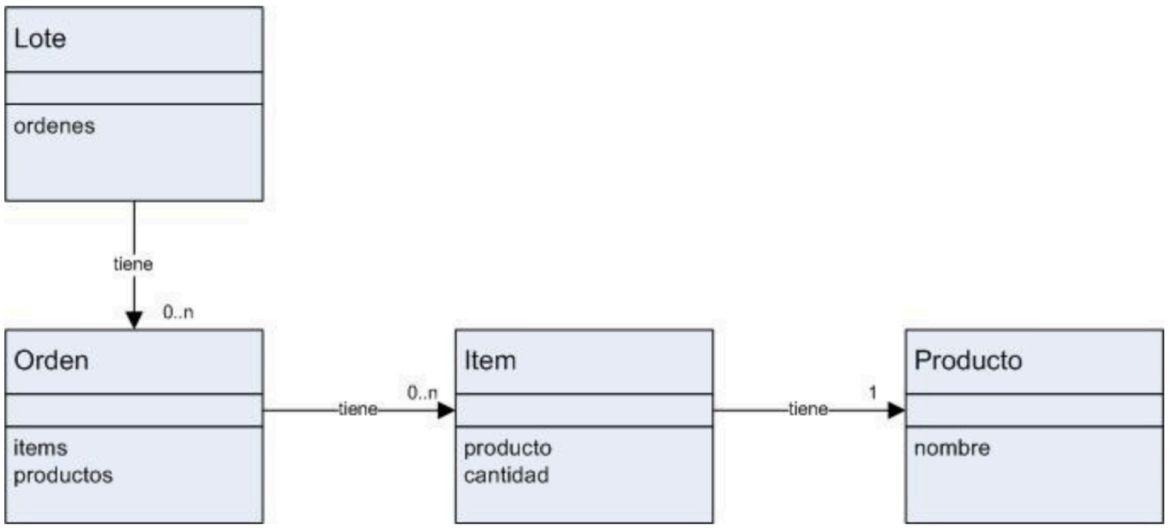
peso en kg * valor de almacenaje del producto por kg

Este valor de almacenaje es distinto para cada producto.
Para los productos comprados hay que agregarle un 20% más, que corresponde al costo de traslado hasta el depósito.

A partir de todo esto, se pide armar los objetos en Wollok que resuelvan:

- Costo total de una orden
- La lista de los productos delicados que aparecen en una orden, ordenada por nombre del producto. Un producto se considera delicado si pesa menos de 5 kg.
- La cantidad total de un determinado producto en un lote (que será la suma de las cantidades en las órdenes que forman el lote y que incluyan ese producto).
- La lista de productos que aparecen en un lote (o sea, aparecen en alguna orden del lote) ordenada por la cantidad total del producto en el lote.
- Implementar el método productos en la clase Orden según el diagrama que se adjunta abajo.

Partir del diagrama que sigue, agregando métodos a las clases que aparecen y agregando las clases adicionales que se necesiten.



Ejercicio 5 - Aprobación de gastos

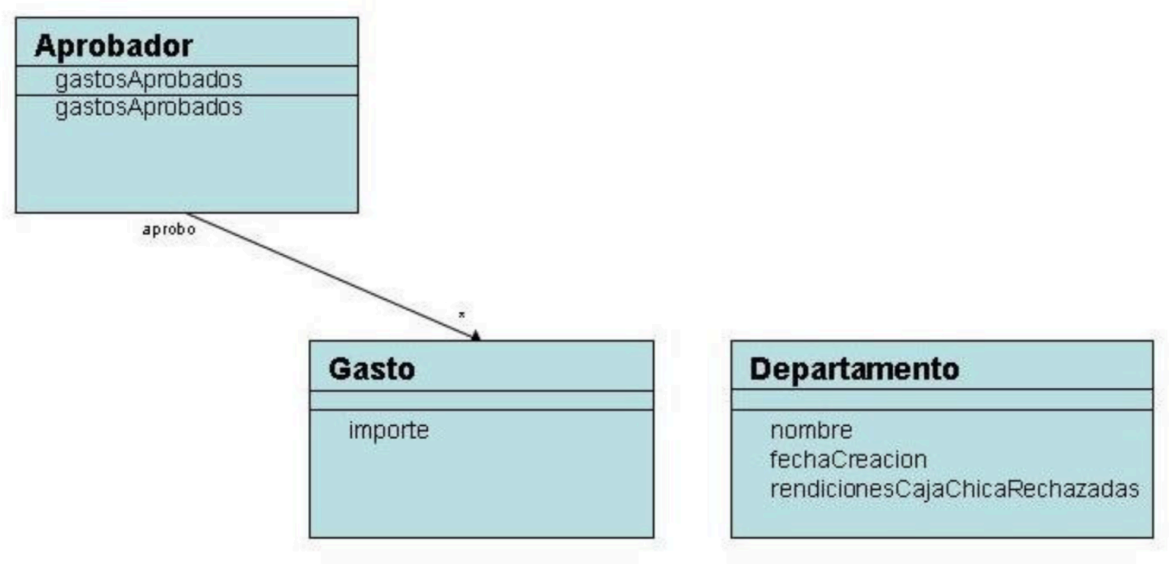
Los gastos de una empresa deben ser aprobados; hay algunos empleados de la empresa que tienen la responsabilidad de aprobar gastos.
Hay dos restricciones a la aprobación de gastos por parte de un aprobador:

- A cada aprobador se le asignan un conjunto de departamentos de la empresa, sólo puede aprobar los gastos de esos departamentos.
- A cada aprobador se le asigna un importe máximo que puede aprobar por día; por lo tanto, cuando se aprueba un gasto debe registrarse la fecha de aprobación.

Hay tres tipos de gastos, que difieren en cómo se determina el departamento al cual pertenecen:

- rendiciones de caja chica: para cada rendición se indica el departamento.
- abonos: todos pertenecen al mismo departamento.
- gastos espciales: pertecen al departamento donde trabaja la persona que solicitó el gasto.

Se parte del siguiente diseño parcial



Consignas

Completar este diseño con los métodos y nuevas clases que hagan falta de forma que se pueda:

- preguntar el importe total aprobado por un aprobador en un día (se cuenta la fecha de aprobación de cada gasto).
- preguntar si un aprobador puede aprobar o no un gasto, dadas las restricciones que se indican arriba.

- realizar la acción de aprobar un gasto por un aprobador, que incluye asignarle la fecha de aprobación al gasto y agregarlo entre los gastos aprobados por el aprobador.
- realizar la acción de rechazar un gasto por un aprobador. El gasto queda marcado como rechazado. Además hay acciones adicionales que dependen del tipo de gasto:
- las rendiciones de caja chica rechazadas deben registrarse de forma tal que el departamento que la rendición tiene asignada resuelva correctamente el mensaje **`rendicionesCajaChicaRechazadas()`**.
- sobre los abonos no se toma ninguna acción.
- para los gastos especiales se registra la fecha de rechazo y se registra de forma tal que luego se pueda averiguar el importe total de gastos rechazados para un empleado en un rango de fechas.
- resolver el método **`rendicionesCajaChicaRechazadas()`** en Departamento, y el método que informa el importe total de gastos rechazados para un empleado en un rango de fechas; y los métodos adicionales que se necesiten.

Indicaciones adicionales:

Para el punto 4, observar el mensaje **`seRechazoAprobacion()`** agregado a Gasto. Implementar los métodos necesarios para que todos los gastos entiendan este mensaje.

Ejercicio 6 - Deporte virtual

Como parte de la implementación de un deporte virtual de equipo (una especie de fútbol simplificado), se representan atributos de los jugadores, a los que se asigna un valor numérico (p.ej. se dice que el jugador Pepe tiene 5 en potencia). A partir de estos valores, se obtienen los valores de atributos que se definen para los equipos, y que sirven a la hora de decidir el resultado de un partido virtual.

Los jugadores se dividen en defensores y delanteros. Todos los jugadores tienen los siguientes atributos:

- visión del juego
- visión de los compañeros
- potencia
- habilidad en los pases

Para los defensores se agrega quite, para los atacantes se agrega anotación.

Para los equipos se establecen estos atributos, para cada uno se indica la forma de calcularlo:

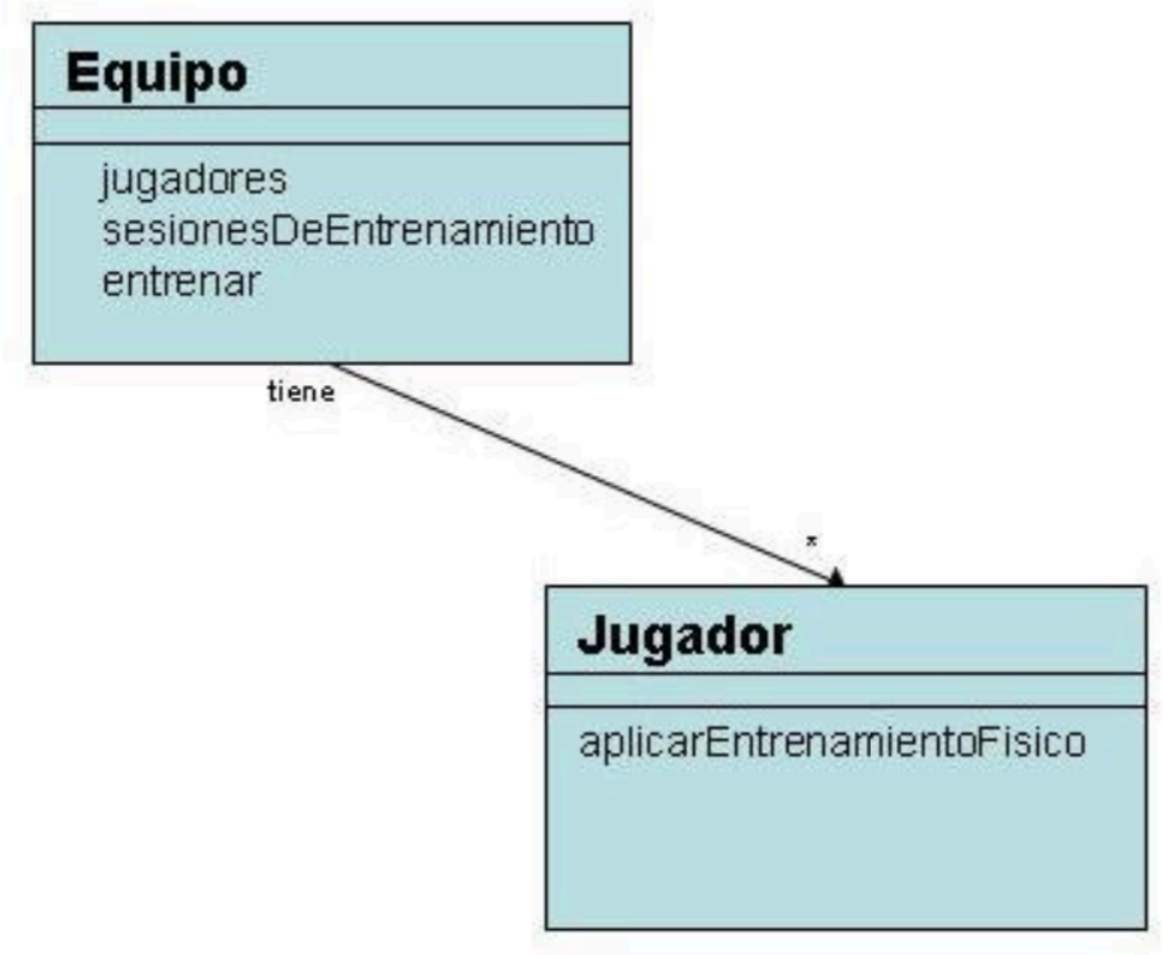
- potencia: suma de la potencia de los dos jugadores más potentes del equipo.
- precisión: suma para cada jugador de (3 * precisión del jugador + pases del jugador)
- La precisión de un jugador se define así: para defensores es su valor de quite, para delanteros es su valor de anotación.
- visión: suma para cada jugador de su valor de visión general.
- La visión general de un jugador se define así: para defensores se suma visión de los compañeros + visión del juego, para delanteros se suman visión del juego + pases.

Cada equipo tiene un conjunto de sesiones de entrenamiento, que modifican los valores de cada uno de sus jugadores. Hay distintos tipos de sesiones de entrenamiento:

- táctico:
    - para defensores sube medio punto de visión del juego y un punto de visión de los compañeros
    - para delanteros sube medio punto de pases, medio de visión del juego, y medio de visión de los compañeros.
- físico: sube un punto la potencia y medio punto los pases; para los defensores además sube medio punto el quite.
- lírico:
    - para los defensores sube un punto de pases y medio punto la visión del juego
    - para los delanteros sube un punto de pases y medio punto la anotación

Cada equipo tiene sus sesiones, que pueden incluir varias del mismo tipo; p.ej. un equipo puede incluir dos sesiones de entrenamiento físico, entonces al entrenar sube dos puntos la potencia de cada jugador, una por cada sesión.
El entrenamiento de un equipo consiste en aplicar cada una de sus sesiones a cada jugador.

Se parte de este diseño:



Y de la implementación de este método:

```
class Sesion {
	cons jugadores

	method entrenar() {
		jugadores.foreach({ jugador =>
			selft.sesionesDeEntrenamiento().foreach({ sesion =>
				sesion.aplicarA(jugador)
			})
		})
	}
}
```

Consignas

Completar este diseño con los métodos y nuevas clases que hagan falta de forma que se pueda:

1. Conocer el valor de potencia, precisión y visión de un equipo
2. Indicar a un equipo que entrene.

Ayudas para el ítem 2

Tener en cuenta el `aplicarEntrenamientoFisico()` agregado a Jugador, implementarlo y agregar los análogos para los otros tipos de sesiones.
OK que queda raro que sea el jugador quien sepa cuáles son los efectos de los tipos de entrenamiento, en vez de saberlo la sesión; simplemente aceptarlo.
Una vez entendido esto, pensar qué debe hacer una sesión de cada tipo al recibir el mensaje `aplicarA(jugador)`

Ejercicio 7 - Administradores de espacio

Una empresa de software quiere desarrollar un administrador de espacio de uso general, que sirva para acomodar cantidades de carga en contenedores.
Dos usos del administrador son:

- distribuir animales entre una flotilla de camiones de ganado.
- distribuir bultos de mercadería en un conjunto de depósitos.

Al crearse el contenedor se le pasa un conjunto de contenedores (p.ej. camiones o depósitos), esos son los que va a manejar.
De cada contenedor (p.ej. camión o silo) se espera poder:

- preguntarle si acepta un determinado ítem. P.ej. a un camión si acepta un animal, o a un depósito si acepta un bulto.
- indicarle que agregue un ítem. P.ej. a un camión que agregue un animal.

Las operaciones que debe soportar el administrador son:

- ubicar un conjunto de ítems entre sus contenedores. Para cada ítem, ubicar un contenedor que lo acepte, y agregarlo

- indicar la capacidad disponible total

Concretamente se pide:

- desarrollar la clase CamionDeGanado. Un camión acepta animales hasta un determinado peso; el peso máximo es el mismo para todos los camiones. Además, todos los animales deben ser de la misma especie y del mismo sexo. O sea, si un camión está vacío puede aceptar cualquier animal; pero si ya tiene p.ej. una vaca hembra, sólo acepta vacas hembra. Además, si p.ej. el peso máximo para los camiones es de 10000 kg y la suma del peso de los animales que ya tiene es 9800 kg, entonces no puede incorporar animales de más de 200 kg.
- Implementar las clases Vaca y Cerdo, a cuyas instancias se pueda preguntar especie, peso y sexo; alcanza con que la especie sea un String (p.ej. "Vaca" para las vacas).
- desarrollar la clase Deposito. Un depósito acepta bultos hasta un determinado volumen.
- Implementar la clase Bulto a la que se le puede preguntar su volumen.
- Implementar el AdministradorDeEspacio con las funcionalidades requeridas. Tener muy en cuenta qué mensajes se le envían al contenedor (camión o depósito). El AdministradorDeEspacio no debería enviarle mensajes a los ítems (animales / bultos).
- Hacer andar el AdministradorDeEspacio con camiones/animales y con depósitos/bultos.
- Para esto tanto camiones como depósitos deben entender los mensajes que le va a enviar el Administrador, si no es así agregar los métodos que hagan falta.

Ejercicio 8 - Combates intergalácticos

Un emperador intergaláctico necesita un software para hacer combatir su flota de naves espaciales. Las naves se agrupan en escuadrones, y los combates ocurren entre escuadrones.
Se pide modelar en Wollok el sistema que permita atacar un escuadrón enemigo con un escuadrón de naves. Esto implica resolver la mecánica de combate (ver especificaciones más abajo) e informar del resultado del mismo. La implementación debería icluir algo como:

```
class Escuadro {
	const naves

	method ataca(escuadron) {
		naves.foreach({ nave => nave.ataca(escuadron})
	}
}
```

Características de las naves:

Existen 4 tipos de naves: cazador, crucero, nave de batalla y destructor. Todas las naves tienen 3 características: poder de ataque, integridad del casco y escudo. Además, cada nave tiene distinta cantidad de cañones, por lo que cada tipo de nave ataca más o menos veces en cada ronda de combate:

Nave	Ataque	Casco	Escuto	Ataque x Ronda
Cazador	100	1000	10	1
Crucero	400	13000	50	3
Nave De Batalla	1200	60000	200	10
Destructor	3000	150000	500	25

Los destructores disponen de un cañón ultra-cósmico que dispara un rayo re-galáctico, capaz de destruir cualquier nave con un sólo impacto. Este cañón se puede usar sólo una vez por combate, y en el turno en que se usa, el destructor pierde sus escudos (o sea su escudo queda en 0) y no puede disparar su armamento convencional.
Los destructores deciden usar el cañón ultra-cósmico cuando atacan a otro destructor o a una nave de batalla, obviamente si ya no fue usado en el combate.
Los cruceros, al ser naves experimentales, tienen en cada turno un 25% de probabilidad de que su armamento no funcione, y por lo tanto que no realice ningún ataque en ese turno.

Mecánica de combate

Durante un combate, todas las naves de un escuadrón se enfrentan a las del escuadrón oponente en rondas, con la siguiente mecánica:

Primero ataca el defensor y luego el atacante.

Las naves de un escuadrón atacan siempre empezando por la más débil (la que tiene poder de ataque más bajo) y así sucesivamente. Si dos naves tienen el mismo poder de ataque no importa el orden de ataque.
Cada ataque de una nave consiste en elegir un objetivo al azar del escuadrón enemigo y atacarlo con todo su poder de ataque. Primero tiene que vencer el escudo del objetivo, y la diferencia restante es lo que llega al casco del objetivo. El escudo perdido se regenera al finalizar cada ronda; pero el casco de una nave sólo se regenera una vez finalizado el combate.
Las excepciones a esta regla de desgaste y regeneración son los cazadores, cuyo casco nunca se regenera.

Si una nave pierde todo su casco, dicha nave explota. Las naves que explotaron se eliminan de los escuadrones al finalizar la ronda, porque aunque hayan explotado se considera que dispararon contra los enemigos antes de recibir daño.
Un combate finaliza cuando uno de los dos escuadrones combatientes pierde todas sus naves.

Dificultades

- Elemento de aleatoriedad para la determinación de objetivos y de eventos como el cañón ultra- cósmico y el ataque de los cruceros.
- Que en una ronda de combate ataquen todas las naves, independientemente de si son o no destruídas en la misma.

Ayuditas

- Pensar cuándo conviene usar VC y MC, y cuándo VI y MI.
- Pensar que existe una clase Batalla que se encarga de resolver el combate, e informar el resultado. ¿Tiene sentido que exista un objeto instancia de Batalla en todo momento?
- ¿Vale instanciar Batalla únicamente por la duración del combate?

Para los que tengan ganas

Pensar qué clases agregar/quitar/modificar para poder hacer que los escuadrones estén en órbita de algún planeta, siendo que un conjunto de planetas pertenecen a un emperador y un planeta puede tener sólo un escuadrón o no tener uno. Un emperador ataca planetas, no escuadrones, y un planeta sólo puede ser atacado si no pertenece al mismo emperador y si sí tiene un escuadrón en órbita.

Ejercicio 9 - Planetas

Una galaxia está dividida en imperios; cada imperio controla un conjunto de planetas de los que extrae recursos.
Además, cada imperio tiene naves-fábrica robotizadas que viajan por el espacio, y que también producen recursos.
Periódicamente el imperio le "solicita" a cada productor (planeta o nave-fábrica) que haga una remesa de los recursos que puede producir hacia los depósitos centrales.

Dentro de los planetas, se distinguen distintos tipos que producen distintos recursos. A continuación se describen los tipos de planetas y cuánto producen cada vez que se lo "solicita" el imperio:

- los planetas agrícolas producen trigo en esta cantidad:

  producción por habitante * población del planeta

  donde la producción por habitante es la misma para todos los planetas agrícolas del universo.
- los planetas minerales producen hierro en esta cantidad:

  capacidad de producción inicial / (edad * coeficiente de desgaste)

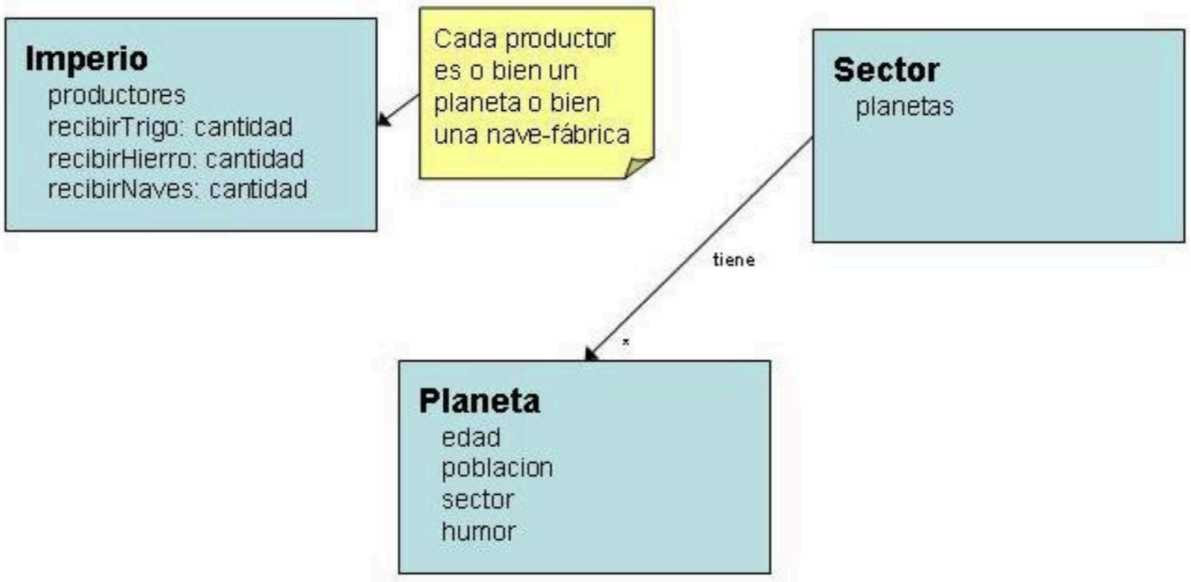
  en donde la capacidad de producción inicial es particular de cada planeta, y el coeficiente de desgaste depende del sector de la galaxia en donde esté el planeta (p.ej. el sector de Sirio tiene un coeficiente de 5, mientras que el de Orión tiene 0.35; por lo tanto las minas de los planetas de Sirio se desgastan más rápido).

- los planetas industriales tienen fábricas que producen naves espaciales; cada fábrica produce esta cantidad:

  capacidad máxima de producción * humor del planeta.

  en donde el humor es un factor entre 0 y 1. La capacidad máxima de producción es particular para cada fábrica. El producido del planeta es la suma del producido de sus fábricas.

Las naves-fábrica producen naves espaciales, en una cantidad particular de cada nave-fábrica, y que no es alterada por ningún factor (o sea, si la capacidad de producción de una nave-fábrica es 10, cada vez que le soliciten una remesa producirá exactamente 10 naves).
Se parte del siguiente diseño



Y de esta implementación

```
class Imperio {

	method solicitarRemesaProdiccion() {
		productos.foreach({ producto =>
			producto.enviarProduccion(self)
		})
}
```

Consignas

Hay que:

- agregar lo que haga falta (más métodos en las clases del diagrama, nuevas clases) para que funcione `solicitarRemesaProduccion()` (o sea, para que todos los productores entiendan `enviarProduccion(...)`).
- implementar el método planetasContaminados en la clase Sector, y los otros métodos que hagan falta. Un planeta está contaminado si tiene más de 500 años y su población supera el millón de personas.
- implementar lo que haga falta para determinar el productor (planeta o nave-fábrica) más valioso de un imperio. El valor de un planeta se calcula así:

poblacion / (20 + edad)

y el de una nave-fábrica es

capacidad de producción * 50

Para los planetas minerales hay que sumar un 30% al valor, porque son los más escasos.

Aclaraciones

- sí vale que cada fábrica conozca a su planeta y que también el planeta conozca a sus fábricas.
- no vale que ni los planetas ni las naves-fábrica conozcan a su imperio.
- no tiene sentido preguntarle ni la edad, ni la población, ni el humor, ni el sector a una nave-fábrica.

Ejercicio 10 - Hipódromo

Un hipódromo nos pide construir una aplicación que maneje los resultados de las carreras, el procesamiento y pago de apuestas, y algunas otras cuestiones.
Cada día de carreras se lleva a cabo lo que se llama una reunión, en la que se corren varias carreras. En cada carrera compiten varios caballos, que deben inscribirse previamente.
De los caballos inscriptos, algunos corren la carrera y otros no.
El resultado de una carrera se define por qué caballo ocupó qué puesto, ej. para una carrera en la que corrieron 5 caballos:

- Akiro
- Séneca
- Calander
- Trinchete
- Pacotino

Se hacen apuestas referidas al resultado de las carreras, cada apuesta corresponde a una carrera. Hay tres tipos de apuestas:

- Apuesta a ganador.
- Apuesta perfecta, en la que el apostador indica qué caballo saldrá primero, qué caballo saldrá segundo y qué caballo saldrá tercero.
- Apuesta sencilla, en la que el apostador indica un solo caballo, y gana si el caballo sale primero, segundo o tercero.

El otro dato que importa de la apuesta es el importe.

Ejemplos de apuestas

- apuesta a ganador para Trinchete para la tercera carrera, de 10 pesos.
- apuesta perfecta para la tercera carrera: ganador Séneca, segundo Akiro, tercero Trinchete; de 15 pesos.

Como no haremos análisis relacionados con los caballos apostados, los podemos representar por su nombre.

¿Cómo se calcula cuánto pagar por una apuesta? El cálculo básico depende del tipo de apuesta:

- las apuestas a ganador pagan 2.2 veces la apuesta.
- las apuestas perfectas pagan 5 veces la apuesta.
- las apuestas sencillas pagan 1.4 veces la apuesta.

A partir de esta cuenta se hacen ajustes que dependen de si se trata de una carrera normal o de un gran premio:

- para las carreras normales, hay un tope de pago de $1000 pesos por apuesta.
- para los grandes premios no hay tope, pero se paga un 5% menos.

Consignas

A. Modelar este dominio de forma tal de poder:
   1. Inscribir los caballos en las carreras, y saber qué caballos están inscriptos para una carrera.
   2. Cargar el resultado de una carrera, o sea qué caballo ocupó cada lugar.
      Poder consultar luego esta información, p.ej. para mostrarla en los paneles y pantallas del hipódromo, y también para el procesamiento de apuestas (como veremos).
   3. Saber qué caballos faltaron a una carrera, o sea se inscribieron pero no corrieron.
   4. Conocer el importe total en apuestas hechas para una carrera, y la apuesta más alta para una carrera.
   5. Saber si una apuesta es o no ganadora. Una apuesta para una carrera de la que todavía no se cargó el resultado no se considera ganadora.
   6. Saber cuánto hay que pagar por una apuesta, de acuerdo al cálculo descripto. Para las apuestas no ganadoras se paga 0 pesos.

Agregar a lo anterior el manejo de caja. La caja paga las apuestas ganadoras, y también los premios de sorteos que se llevan a cabo durante la reunión. Los números para los sorteos se regalan a los asistentes que hacen compras en las tiendas y restaurantes del hipódromo. Los sorteos tienen tres números ganadores, para cada sorteo se establecen los importes a pagar para primer, segundo y tercer premio.

B. Se pide específicamente poder:
   1. Registrar el resultado de un sorteo, que consiste en indicar los números que salieron en primer, segundo y tercer lugar.
   2. Saber cuánto falta pagar en apuestas y/o sorteos. Para esto hay que
      registrar el pago de una apuesta ganadora y/o número ganador de sorteo.
      registrar en la caja todas las apuestas y números de sorteos repartidos. Los únicos sorteos ganadores son los que corresponden a sorteos de los que ya se registró el resultado.
   3. Saber cuál es la ganancia del día, que es la suma del importe de cada apuesta menos el importe a pagar en apuestas y sorteos.
      Tener en cuenta sólo las apuestas/números de las carreras/sorteos ya realizados.

Hay algún objeto que se agrega para el punto ‘B’. que conviene que sea polimórfico con algún otro introducido en el punto ‘A’., ¿cuáles son estos objetos que conviene tratar

polimórficamente?

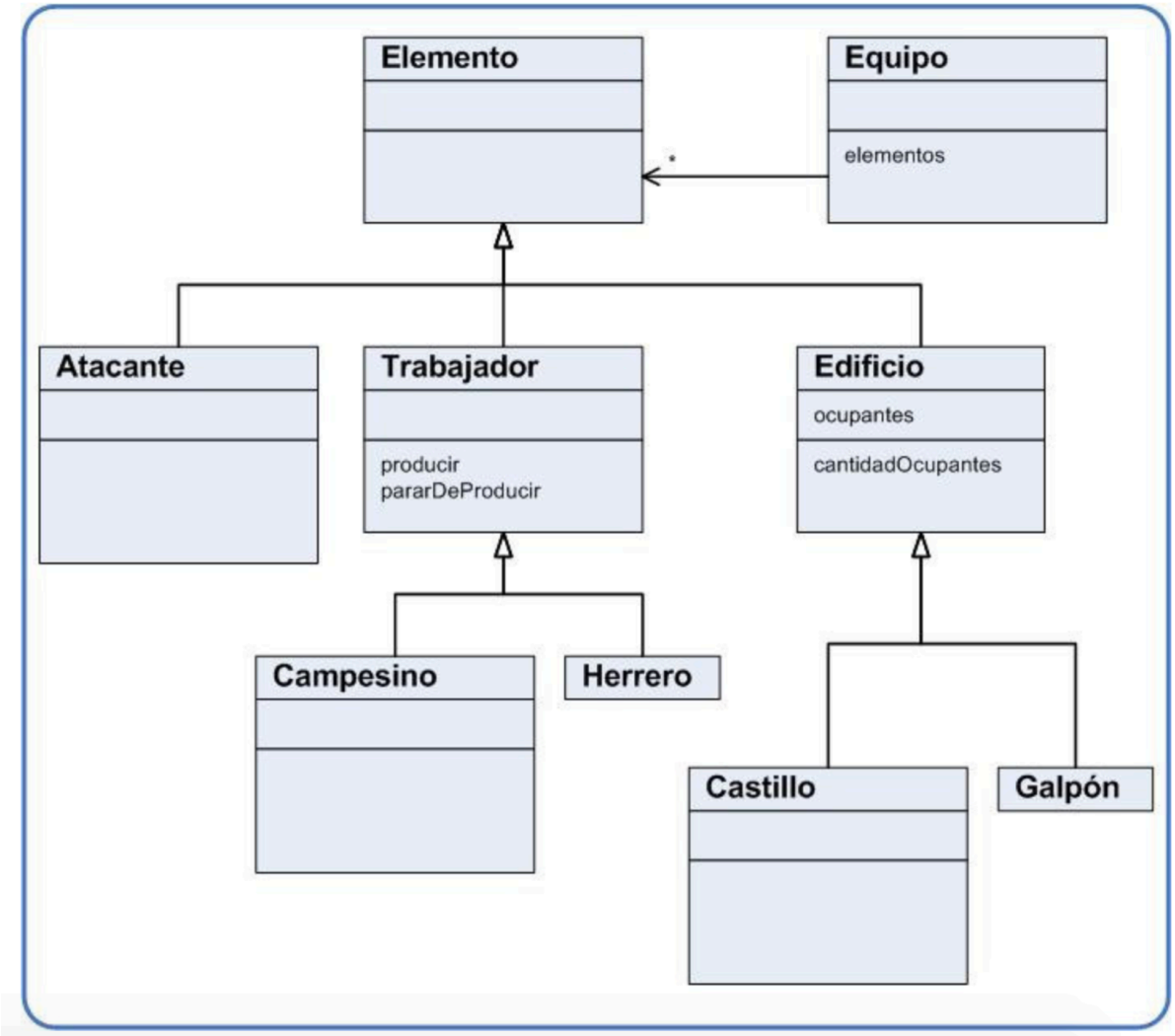
Ejercicio 11 - Juego de estrategia

Planteo general

En un juego de estrategia, que ya está en desarrollo, se quiere agregar el modelado de la dinámica de combate entre elementos.
Lo que está hecho hasta ahora resuelve

- el modelo de un equipo y los elementos (personas, edificios, etc.) que lo componen.
- la producción de los bienes que necesita el equipo, de esto se encargan algunos de los elementos agrupados en la superclase Trabajador.
- el modelo de edificios, un edificio puede ser ocupado y eso tiene consecuencias que escapan a lo que tenemos que hacer ahora.

El modelo de lo construido hasta ahora puede verse en este diagrama, al que se le agrega la clase Atacante que (ya veremos) nos será de utilidad para lo que tenemos que resolver.



Planteo - dinámica de combate

En cada combate hay un elemento atacante y uno defensor. No se admiten ni ataques ni defensas grupales.

Sólo los guerreros y los cañones pueden atacar, y sus blancos pueden ser solamente campesinos y castillos. El resto de los elementos no puede ni atacar ni ser atacado.
Los ataques se resuelven a partir del poder ofensivo del atacante y del poder defensivo del atacado. A partir de un ataque se produce un desgaste tanto en el atacante como en el atacado. El poder inicial y su desgaste se computan según lo que sigue:

Poder ofensivo:

- guerrero: el poder ofensivo de un guerrero es en cada momento el de su arma más potente. Si no tiene armas, su poder ofensivo es 0.
  Cada guerrero puede tener varias armas; entre las armas consideraremos las espadas y las ballestas. Un guerrero puede tener más de una espada, y también más de una ballesta.
- cañón: el poder ofensivo inicial es el mismo para todos los cañones. El desgaste es de un punto por ataque que realiza.
- espada: el poder ofensivo inicial se asigna a cada espada cuando se la crea. Las espadas nunca se desgastan.
- ballesta: el poder ofensivo de una ballesta es su cantidad de flechas / 2. En cada ataque pierde 3 flechas.

Poder defensivo:

- campesinos: el poder defensivo es 10 si nunca fue atacado, y 0 si fue atacado alguna vez.
- castillos: el poder defensivo inicial se asigna a cada castillo cuando se lo crea. El desgaste en cada ataque sufrido es un punto por cada 10 puntos de poder ofensivo del atacante.

Los castillos también pierden un punto de poder defensivo cada 10 años de edad. Como es el primer caso donde interviene la edad de un elemento, también hay que modelar edad y envejecimiento. Esto hay que hacerlo para todos los elementos (tanto los que pueden ser atacantes o defensores como los que no), más adelante se usará en otras funcionalidades.

Consigna

Armar la estructura de objetos que modele lo recién descripto tal que se pueda:

- indicarle a un atacante que ataque efectivamente a un defensor. En este caso deben actualizarse los poderes de atacante y defensor según las reglas de desgaste. Primero se desgasta el defensor, y después el atacante.
- indicarle a un equipo que envejezca, o sea sumarle uno a la edad de cada uno de sus elementos.
- preguntarle a un equipo cuáles son sus elementos viejos; un elemento se considera viejo si su edad es mayor a 30.

- preguntarle a un atacante si le conviene atacar a un defensor.
  El criterio general es: a un atacante le conviene atacar a un defensor si el poder ofensivo del primero es mayor al poder defensivo del segundo. Este criterio vale para los futuros atacantes que se prevé agregar al juego.
  En particular para los cañones, además del caso anterior siempre les conviene atacar si son viejos.
- preguntarle a un juego cuáles son los elementos inservibles, en donde
- un atacante es inservible si su poder ofensivo es 0.
- un campesino es inservible si su poder defensivo es 0.
- el resto de los elementos nunca llegan a ser inservibles.

Además

Indicar

- qué tipos identificás en los objetos que armaste, y para cada uno
- qué mensaje/s incluye
- qué objetos pertenecen
- qué objeto/s lo usa/n
- qué delegación hay en el cálculo de poder ofensivo de un atacante.

Ejercicio 12 - Librería

1. En cierto lugar de la ciudad se establece una librería que vende libros, y también colecciones anuales de revistas.
   Una colección anual de revistas se compone de todos los fascículos de la revista aparecidos en un año, p.ej. El Gráfico 1987, o .code 2004.
   El precio de una colección se calcula así:
   (precio x fascículo cantidad de fascículos) + precio de encuadernación
   donde:
   - el precio de encuadernación es el 20% de (precio x fascículo cantidad de fascículos), con unmínimo de $10.
   - el precio por fascículo se establece para cada revista sin importar su antigüedad, p.ej. se establece que cada fascículo de El Gráfico se va a vender a $12, entonces este precio vale (p.ej.) tanto para El Gráfico 2002 como para El Gráfico 1938.

   El precio de cada libro es informado por algún usuario, o sea que para obtenerlo no hace falta ninguna cuenta.

   ¿Qué pasa si un cliente quiere pagar al contado y pide pagar menos?
   - Para las colecciones no hay descuento por pago contado, o sea que es el precio normal.
   - Para los libros sí se hacen descuentos por pago contado, que dependen de si se trata de un libro nuevo o usado;
   - Para los nuevos se hace el mismo porcentaje que la editorial del libro le hace a la librería por pago contado (este porcentaje es único para cada editorial), mientras que para los usados el porcentaje es el mismo para todos (p.ej. el encargado dice "ahora para los usados el descuento x pago contado es 25%").

   Modelar usando objetos esta librería de forma tal de poder responder lo siguiente:
   - en qué ítems de venta (libros y/o colecciones) aparece un determinado autor. Para las colecciones se consideran los autores que hayan colaborado en al menos uno de los fascículos.
   - el precio de un ítem.
   - el precio contado de un ítem.
   - a colección de ítems de venta ordenados por nombre. Para las colecciones se considera el nombre de la revista seguido del año, p.ej. "El Gráfico 1998".

   Tener en cuenta que las colecciones de distintos años de la misma revista no necesariamente tienen la misma cantidad de fascículos, las revistas pueden cambiar de frecuencia, o puede haber fascículos especiales.

2. Agregar al modelo los objetos necesarios para registrar las ventas que hace la librería. Cada venta se refiere a un solo ítem, p.ej. si un cliente se lleva dos libros y una colección se consideran tres ventas distintas.
   Se debe poder responder lo siguiente:
   - monto total vendido en un determinado día.
   - los días en los que se registró alguna venta.

Para cada venta hay que registrar si se hizo al contado o no, porque de eso depende el precio que el cliente pagó por el ítem.
También hay que tener en cuenta las ventas telefónicas. Si una venta es telefónica, al precio se le suma el costo de envío, que es de un 5% sobre el precio del ítem para esa venta.

3. Agregar al modelo los objetos necesarios para registrar las reservas que toma la librería. No nos importa mantener el stock, solamente saber qué reservas se hicieron.
   Cada reserva corresponde a un solo ítem, análogo a las ventas.
   Las reservas se vencen a los 6 días, excepto las marcadas como especiales, que vencen a los 14 días. Estos dos números (6 y 14) son políticas de la empresa que pueden cambiar, y ese cambio debe poder ser incorporado al sistema sin necesidad de tocar código.
   Se debe poder responder:
   - Las reservas que vencen hoy, que (claramente) no incluyen ni las canceladas ni las resueltas (ver abajo).
   - Registrar la postergación de una reserva. Las reservas comunes se postergan una sola vez y por 3 días, las especiales hasta 2 veces y 7 días cada vez.
   - Registrar la cancelación de una reserva.
   - Conocer el monto total de ventas de un día que provinieron de una reserva. Para eso, en el momento de hacerse la venta el que la registra indica de qué reserva provino, y esa reserva se considera resuelta.
   - Conocer, para un determinado día (del pasado), cuántas reservas hechas ese día se cancelaron, cuántas se resolvieron y cuántas siguen vigentes.