Ejercicio
Carga de Camiones

Dr. Julio Castillo

Caso de estudio: Carga de camiones

Una empresa de transporte de cargas necesita un software que la ayude a organizarse con la carga de las camiones que maneja. La empresa puede recibir packings, cajas sueltas y bidones que transportan líquido.

- Un packing es una estructura de madera que arriba tiene un montón de cajas, se envuelve todo con plástico para que no se desbanden las cajas. Todas las cajas tiene el mismo peso. El peso de un packing es (peso de cada caja * cantidad de cajas) + peso de la estructura de madera que va abajo. Para cada packing se informa qué llevan las cajas, p.ej. Material de construcción.
- De cada caja suelta se informa el peso individualmente, son todas distintas.
- El peso de un bidón es su capacidad en litros por la densidad (o sea, cuántos kg pesa un litro) del líquido que se le carga. Los bidones van siempre llenos hasta el tope.

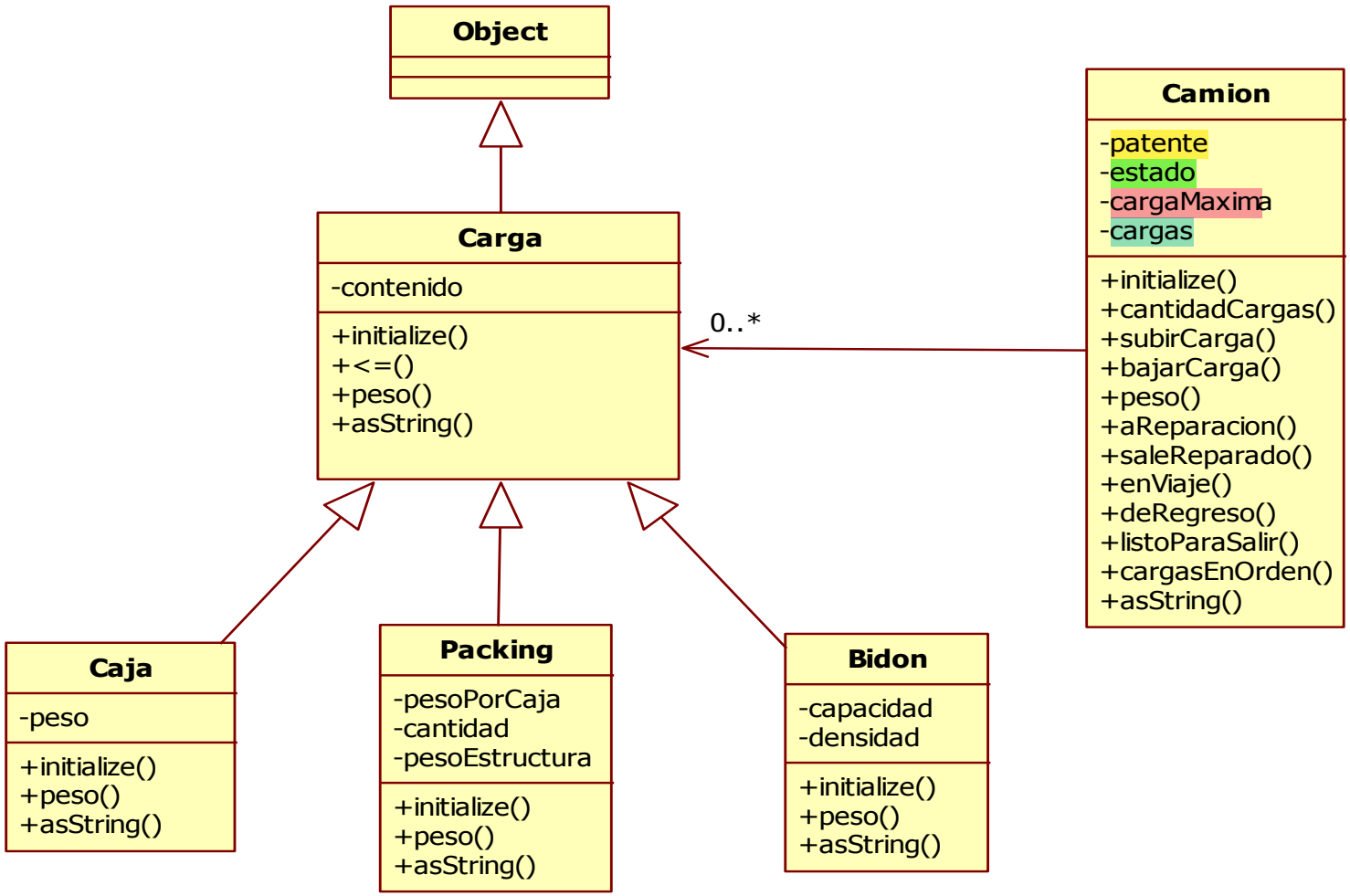
Cada camión puede llevar hasta una **carga máxima** medida en kg. Además, cada camión puede: **estar disponible** para la carga (en cuyo caso ya puede tener cosas cargadas)**, estar en reparación, o estar de viaje.**

Propuesto por: Ing. Martín Polliotto

Requerimientos:

- **Subir** una **carga** al camión, donde la carga puede ser un packing, una caja suelta, o un bidón. Considerar que no se puede saturar un camión con más peso de lo que su carga máxima permite.
- **Bajar** una carga del camión, siempre que el camión se encuentre disponible con cargas y que a su vez la carga se encuentre presente dentro de él.
- Conocer el **total de cargas** de un camión en todo momento y su **peso**.
- Permitir **modificar el estado** de un camión, sea porque entro en reparación, salió de reparación, está en viaje o de regreso.
- Obtener el listado de cargas contenidas **ordenado por peso.**
- Determinar si un camión está **listo para partir,** que es: si está disponible para la carga, y el peso total de lo que tiene cargado es de al menos **75% de su carga máxima.**

Propuesto por: Ing. Martín Polliotto



Para mayor simplicidad, los métodos de acceso fueron omitidos del diagrama

Propuesto por: Ing. Martín Polliotto

Solución

```
Object subclass: #Carga
    instanceVariableNames: 'contenido'
    classVariableNames: ''
    category: 'EjercicioRepaso_Camiones‘

Carga subclass: #Bidon
    instanceVariableNames: 'capacidad densidad'
    classVariableNames: ''
    category: 'EjercicioRepaso_Camiones’

Carga subclass: #Caja
    instanceVariableNames: 'peso'
    classVariableNames: ''
    category: 'EjercicioRepaso_Camiones’

Carga subclass: #Packing
    instanceVariableNames: 'pesoPorCaja cantidad pesoEstructura'
    classVariableNames: ''
    category: 'EjercicioRepaso_Camiones'
```

```
Object subclass: #Camion
        instanceVariableNames:
'patente estado cargaMaxima cargas'
        classVariableNames: ''
        category:
'EjercicioRepaso_Camiones'
```

Clase Bidón

```
capacidad
    ^ capacidad

capacidad: anObject
    capacidad := anObject

densidad
    ^ densidad

densidad: anObject
    densidad := anObject

initialize
    super initialize.
    capacidad :=0.
    densidad :=0.
peso
    ^(capacidad *densidad).

asString
    |cad|
    cad:=super asString, 'Bidon con capacidad: ', capacidad asString, ' y densidad: ',
    densidad asString, '-'.
    ^cad.
```

Clase Caja

```
peso
    ^peso.

peso: anObject
    peso := anObject

initialize
    super initialize.
    peso:=0.

asString
    |cad|
    cad:=super asString, 'Peso de la caja: ', peso asString, '-'.
    ^cad.
```

Clase Packing

```
pesoEstructura: anObject
    pesoEstructura := anObject
cantidad
    ^ cantidad
cantidad: anObject
    cantidad := anObject
```

```
pesoPorCaja
    ^ pesoPorCaja
pesoPorCaja: anObject
    pesoPorCaja := anObject
pesoEstructura
    ^ pesoEstructura
```

```
initialize
    super initialize.
    pesoPorCaja:=0.
    pesoEstructura :=0.
    cantidad:=0.
peso
    |res|
    res:= (pesoPorCaja * cantidad) + pesoEstructura.
    ^res.
asString
    |cad|
    cad:=super asString, 'Packing con Peso por caja:', pesoPorCaja asString, ',
    cantidad: ', cantidad asString, ', Peso de la estructura:', pesoEstructura
    asString, '-'.
    ^ cad.
```

Clase Camión

```
estado
    ^ estado
estado: anObject
    estado := anObject
patente
    ^ patente
patente: anObject
    patente := anObject
aReparacion
    estado:='en reparacion'.
    cargas:= OrderedCollection new.
cantidadCargas
    ^cargas size.
cargaMaxima
    ^ cargaMaxima
cargaMaxima: anObject
    cargaMaxima := anObject
deRegreso
    estado:='de regreso'.
    cargas:=OrderedCollection new.
enViaje
    estado:='en viaje'.
    cargas:=OrderedCollection new.
```

```
initialize
    super initialize.
    patente:=0.
    estado:='disponible'.
    cargaMaxima:=1000.
    cargas:= OrderedCollection new.
listoParaPartir
^((estado = 'disponible') & ((self peso) >=
(cargaMaxima * 0.75))).
```

Clase Camión(2)

```
asString
   |cad|
   cad:='Camion con patente:', patente asString, '- capacidad:',
   cargaMaxima asString, '-estado:', estado asString, String cr.
   cad:=cad, 'Listado de todas las cargas del Camion:', String cr.
   cargas do:[:x|cad:=cad, x asString, ';'.].
   ^cad.

peso
   |val|
   val:=0.
   cargas do:[:x| val:=val+ x peso].
^val.
```

Clase Camión(3)

```
subirCarga: aCarga
    |val msg|
    val:=0.
    msg:='La carga no se ha incorporado'.
    ((estado = 'disponible') & (aCarga isKindOf: Carga))
    ifTrue: [ val:= self peso + aCarga peso.  (val <= cargaMaxima) ifTrue: [ cargas add: aCarga.
    msg:='La carga se ha incorporado al camion'.] ].
    ^msg.
cargasEnOrden
    |cad cargas_ord|
    cad:='Listado de Cargas en Orden de Peso:', String cr.
    cargas_ord:= cargas asSortedCollection.
    cargas_ord do:[:x| cad:= cad, x asString, String cr.].
    ^cad.
bajarCarga: aCarga
    |car msg|
    msg:='Carga no eliminada'.
    car:= cargas detect: [ :x| (x contenido = aCarga contenido)] ifNone: [ car:=nil.].
    ((estado = 'disponible') & (car isNotNil)) ifTrue: [ cargas remove: aCarga.
    msg:='carga eliminada'. ].
    ^msg.
```