Piratas: posible resolución

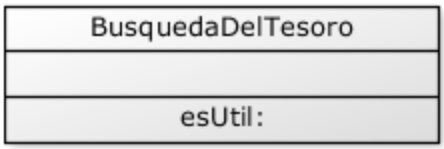
//1. Saber si un pirata es útil para una misión

//Codificación

```
BusquedaDelTesoro >> esUtil: unPirata
```

"Para esta clase de misiones sólo serán útiles como tripulantes aquellos piratas que tengan una brújula, un mapa o una botella de grogXD entre sus ítems, y no tengan más de 5 monedas."

```
^unPirata tenesAlguno: #('brújula' 'mapa' 'botellaDeGrogXD')
      and: [ unPirata cantidadMonedas <= 5 ]
```



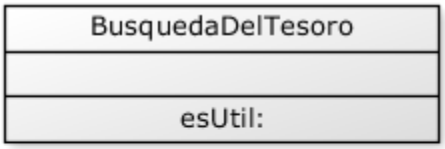

```
Pirata (variables de instancia: items cantidadMonedas)
```

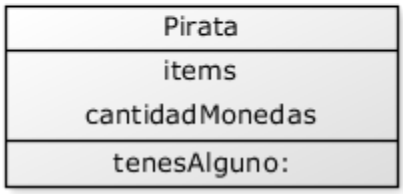
"Se asume que se crean los getters y setters de estas variables"

```
Pirata >> tenesAlguno: itemsNecesarios
      ^itemsNecesarios anySatisfy:
            [ :unItem | self items includes: unItem ]
```

"O bien"

```
      ^self items includesAnyOf: itemsNecesarios
```



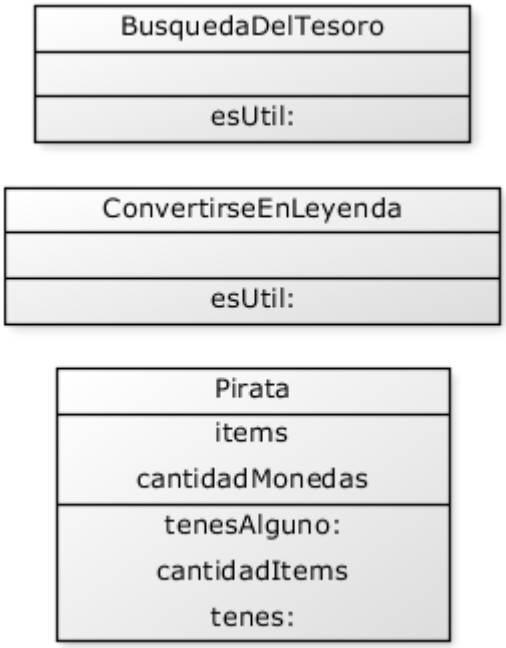



```
ConvertirseEnLeyenda >> esUtil: unPirata
```

"Para que un pirata sea útil en una misión de convertirse en leyenda debe tener al menos 10 ítems, entre los cuales debe figurar un ítem obligatorio definido en cada misión de convertirse en leyenda."

```
      ^unPirata cantidadItems >= 10
            and: [ unPirata tenes: itemObligatorio ]

Pirata >> cantidadItems
      ^self items size
Pirata >> tenes: unItem
      ^self tenesAlguno: (Bag with: unItem)
```

```
"O bien" ^self items includes: unItem
```



Saqueo(definimos una variable de clase CantidadMonedasDeterminada**)**
(definimos una variable de instancia miVictima**)**

Saqueo >> esUtil: unPirata

"Para estas misiones son útiles los piratas que cuenten con menos dinero que una cantidad de monedas determinada (para todas las misiones de saqueo es la misma cantidad, pero se debe poder cambiar) y además se animen a saquear a la víctima de la misión de la que se trate"

```
^unPirata cantidadItems < CantidadMonedasDeterminada
     and: [ unPirata teAnimasASaquear: miVictima ]
```

Pirata >> teAnimasASaquear: unaVictima

"Nos damos cuenta que esto es responsabilidad de la victima => delegamos !!!"

```
^unaVictima seAnimaASaquear: self
```

Barco >> seAnimaASaquear: unPirata

"un pirata tiene que estar *pasado de grogXD* (tener un nivel de ebriedad de al menos 90) como para animarse a saquear a un barco pirata *pasado de grogXD* (tener un nivel de ebriedad de al menos 90) como para animarse a saquear a un barco pirata"

```
^unPirata estasPasadoDeGrogXD
```

Pirata (agregamos la variable de instancia nivelEbriedad**)**

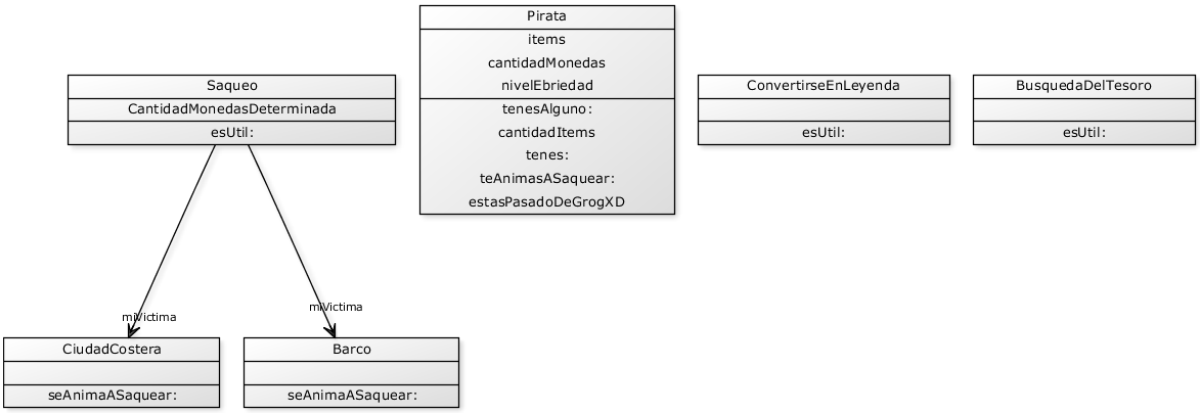
Pirata >> estasPasadoDeGrogXD

```
^self nivelEbriedad >= 90
```

CiudadCostera >> seAnimaASaquear: unPirata

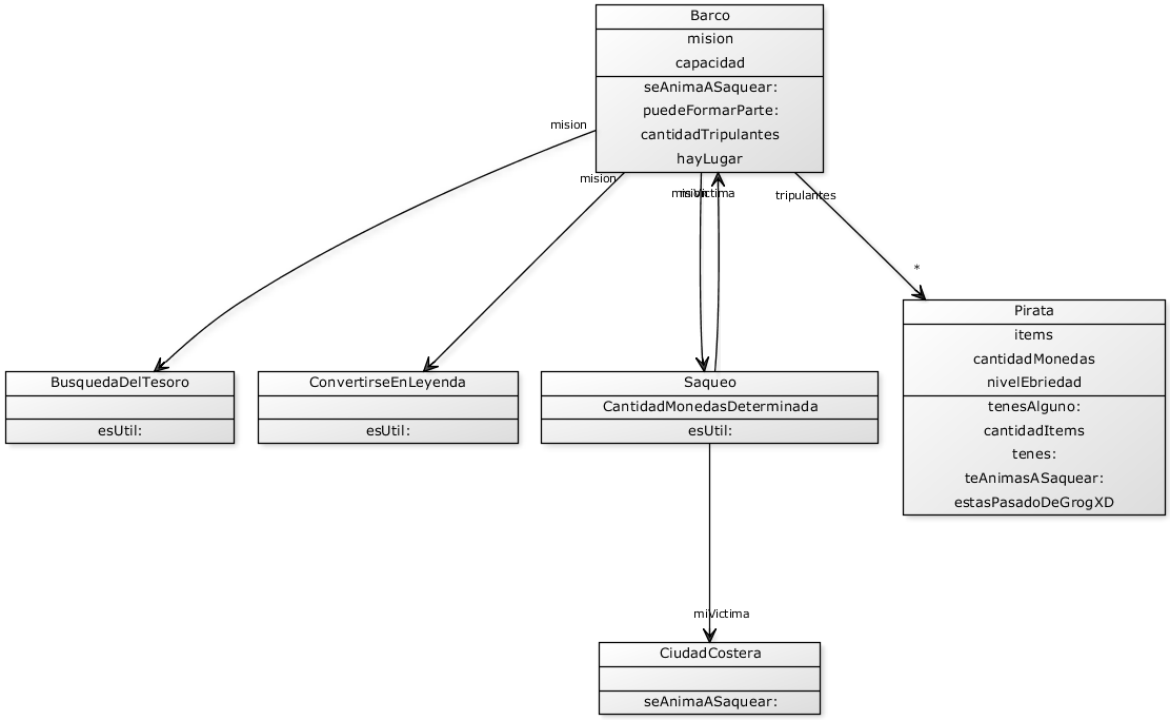
"un pirata debe tener un nivel de ebriedad de al menos 50 para animarse a saquear a una ciudad"

```
^unPirata nivelEbriedad >= 50
```



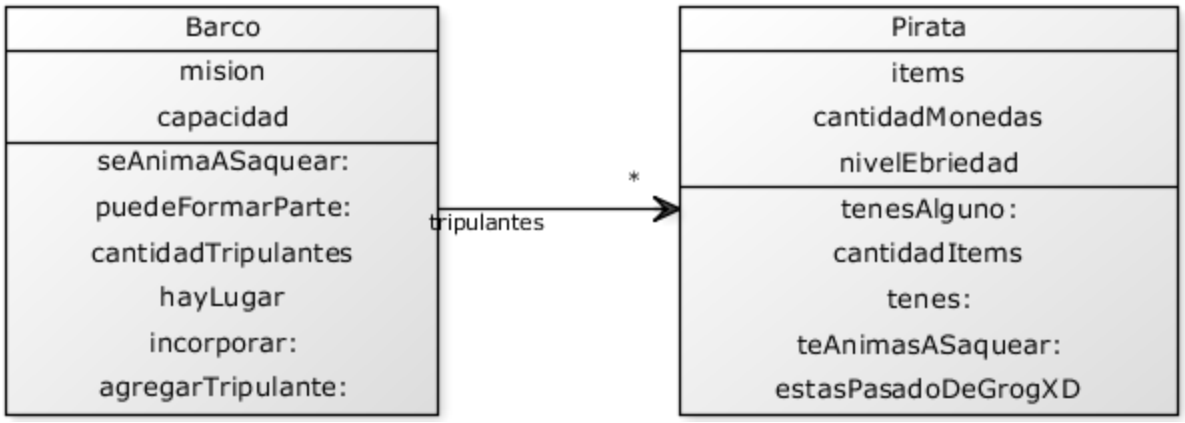
“2a) Saber si un pirata puede formar parte de la tripulación de un barco. Esto ocurre cuando el barco tiene lugar para un tripulante más, y además el pirata sirve para la misión del barco.”

```
Barco (variables de instancias tripulantes mision capacidad)

Barco >> puedeFormarParte: unPirata
	^self hayLugar and: [ mision esUtil: unPirata ]

Barco >> hayLugar
	^capacidad > self cantidadTripulantes

Barco >> cantidadTripulantes
	^tripulantes size
```



2b) Incorporar un pirata a la tripulación de un barco, sólo si esto puede ser llevado a cabo.

```
Barco >> incorporar: unPirata
      ( self puedeFormarParte: unPirata ) ifFalse:
            [ self error: 'No se puede incorporar' ].
      self agregarTripulante: unPirata.

Barco >> agregarTripulante: unPirata
      tripulantes add: unPirata
```



“2c) Hacer que un barco cambie de misión. Al cambiar de misión, el barco echa a los tripulantes que no sirven para su misión actual.”

```
Barco >> cambiarMision: nuevaMision
      tripulantes :=
            tripulantes select: [ :unTripulante |
                  nuevaMision esUtil: unTripulante ]
      self mision: nuevaMision.
```

"O bien"

```
      tripulantes removeAll:
            (tripulantes
                  select: [:unTripulante |
                        (nuevaMision esUtil: unTripulante) not ]).
      self mision: nuevaMision.
```

"O bien … No es prudente eliminar elementos de la colección que se está iterando, por eso se itera sobre una copia de la colección que se obtiene con el mensaje copy"

```
      tripulantes copy do: [ :unTripulante |
            (nuevaMision esUtil: unTripulante)
                  ifFalse: [ self echarA: unPirata ]
            ]
      ].
      self mision: nuevaMision.

Barco >> echarA: unPirata
```

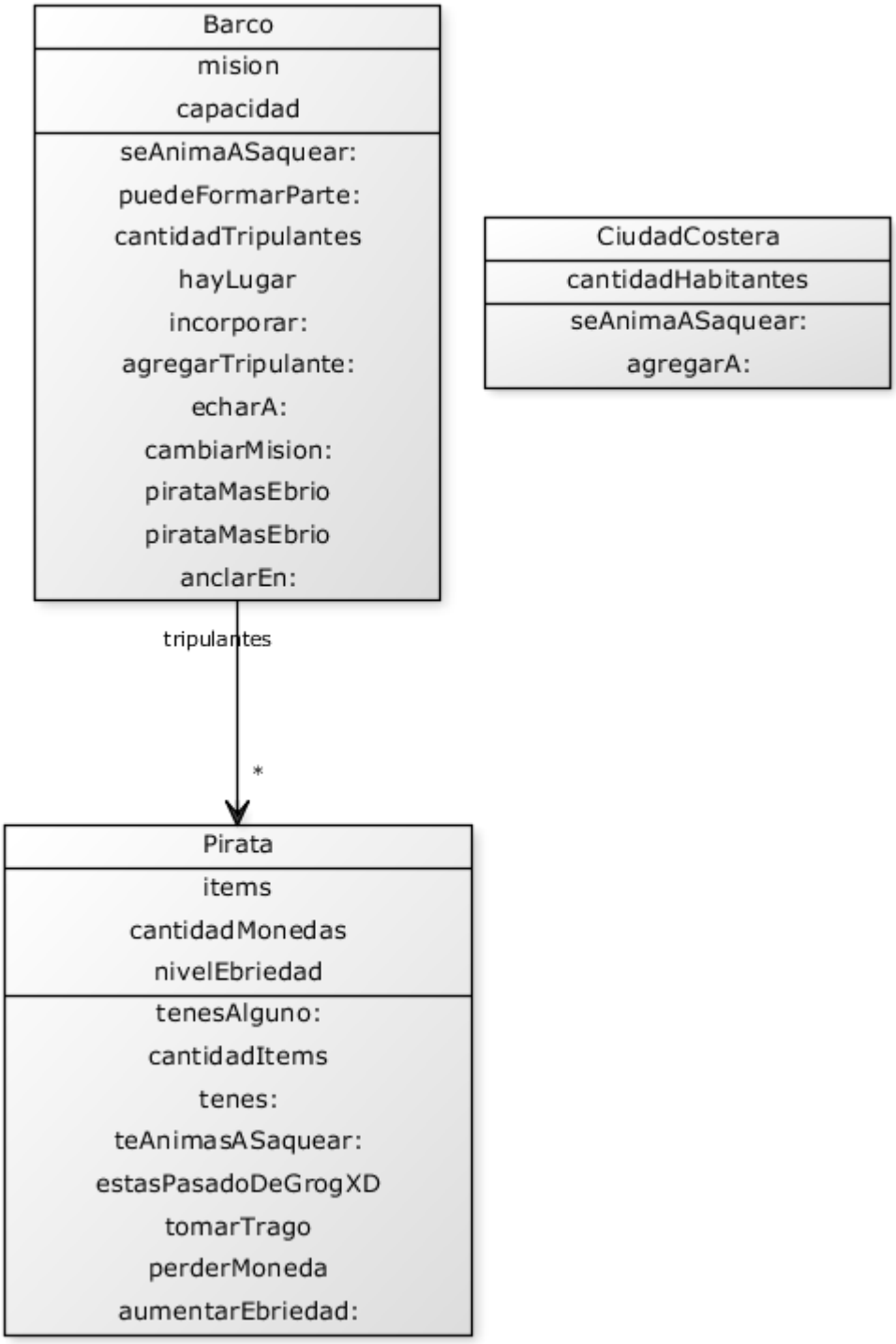
```
	tripulantes remove: unPirata
```

“3a) Saber quién es el pirata más ebrio de la tripulación de un barco.”

```
Barco >> pirataMasEbrio
	^(tripulantes asSortedCollection:
		[ :pirataQueVaAntes :pirataQueVaDespues |
			pirataQueVaAntes nivelEbriedad
				> pirataQueVaDespues nivelEbriedad ]) first
```

“3b) Hacer que un barco ancle en una ciudad costera. Cuando los barcos anclan, toda la tripulación se toma un trago de grogXD, que provoca que el nivel de ebriedad de cada uno suba en 5 unidades y se gasten una moneda.
Además, el más ebrio de los tripulantes del barco queda perdido en la ciudad, o sea que deja de formar parte de la tripulación y la ciudad queda con un habitante más.”

```
Barco >> anclarEn: unaCiudad
	| elMasEbrio |
	tripulantes do: [ :unTripulante |
		unTripulante tomarTrago
	].
	elMasEbrio := self pirataMasEbrio.
	unaCiudad agregarA: elMasEbrio.
	self echarA: elMasEbrio.

Pirata >> tomarTrago
	self perderMoneda.
	self aumentarEbriedad: 5.

Pirata >> perderMoneda
	self cantidadMonedas: self cantidadMonedas - 1

Pirata >> aumentarEbriedad: aumento
	self nivelEbriedad: self nivelEbriedad + aumento

CiudadCostera (variable de instancia cantidadHabitantes)

CiudadCostera >> agregarA: unPirata
	cantidadHabitantes := cantidadHabitantes + 1
```



“4a. Saber si un barco pirata es temible”

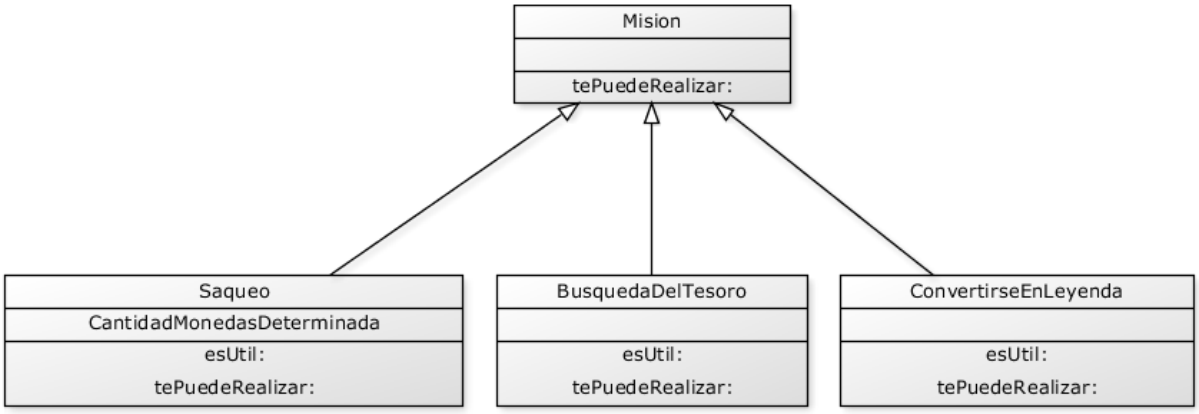
“Vamos a decir que un barco pirata es temible si puede realizar la misión que tiene asignada (asumimos que un barco siempre tiene una misión asignada recordando que esa misión puede cambiar).”

```
Barco >> sosTemible
      ^mision tePuedeRealizar: self
```

“Una misión, para poder ser completada por un barco, debe cumplir ciertos requisitos. El primero, común a todas ellas, es que el barco tenga suficiente tripulación.”

“Nos damos cuenta que necesitamos escribir un comportamiento que es igual para todas las misiones, entonces creamos una clase Mision superclase de Saqueo, ConvertirseEnLeyenda y BusquedaDelTesoro”

```
Mision >> tePuedeRealizar: unBarco
	^unBarco tieneSuficienteTripulacion
```




```
Barco >> tieneSuficienteTripulacion
```

“Se considera que un barco tiene suficiente tripulación cuando la cantidad de sus tripulantes llega al menos al 90% de su capacidad. De cada barco se conoce su capacidad (cuántas personas es capaz de llevar).”

```
	^capacidad * 0.9 >= self cantidadTripulantes
```

```
BusquedaDelTesoro >> tePuedeRealizar: unBarco
```

“Para que una búsqueda del tesoro pueda ser realizada por un barco, al menos uno de sus tripulantes debe tener entre sus ítems una llave de cofre.”

```
	^(super tePuedeRealizar: unBarco) and:
		[ unBarco algunTripulanteTiene: 'llaveDelCofre' ]
```

```
Barco >> algunTripulanteTiene: unItem
	^tripulantes anySatisfy: [ :unTripulante |
		unTripulante tenes: unItem ]
```

```
ConvertirseEnLeyenda >> tePuedeRealizar: unBarco
```

“Esta clase de misiones no tiene condiciones particulares para que pueda ser realizada con un barco, por ende no haría falta definir este método!”

```
	^(super tePuedeRealizar: unBarco) and: [ true ]
```

"Insisto en que podríamos evitar escribir este método y ya"

```
Saqueo >> tePuedeRealizar: unBarco
```

“Para que un saqueo pueda ser realizado por un barco, la víctima debe ser vulnerable al barco”

```
	^(super tePuedeRealizar: unBarco)
		and: [ victima sosVulnerable: unBarco ]
```

“Vemos que las instancias de Barco y de CiudadCostera son polimórficas con respecto a los mensajes `seAnimaASaquear: unPirata y sosVulnerable: otroBarco`"

Barco >> `sosVulnerable: otroBarco`

“un barco pirata es vulnerable a otro si tiene como máximo la mitad de los tripulantes que el que lo quiere saquear”

```
^self cantidadTripulantes <= otroBarco cantidadTripulantes
```

CiudadCostera >> `sosVulnerable: unBarco`

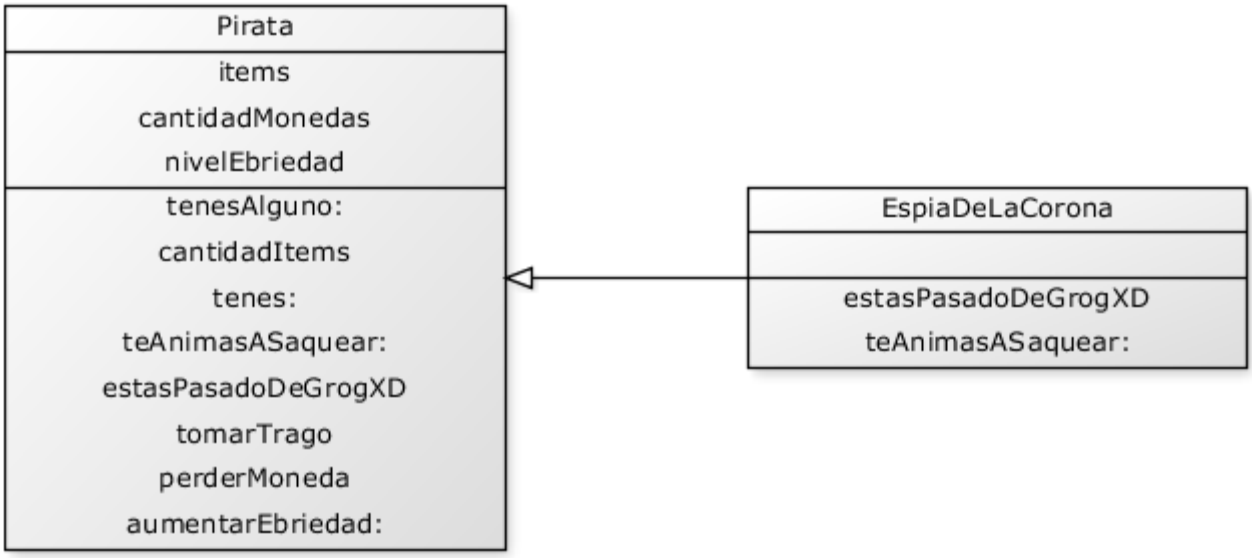
“Una ciudad costera es vulnerable a un barco si la cantidad de tripulantes es al menos el 40% de la cantidad de habitantes de la ciudad, o bien cuando toda la tripulación del barco está pasada de grogXD”

```
^unBarco cantidadTripulantes >=
      (0.40 * self cantidadHabitantes)
            or: [ unBarco todosEstanPasadosDeGrogXD ]
```

Barco >> `todosEstanPasadosDeGrogXD`

```
^tripulantes allSatisfy: [ :unPirata |
      unPirata estasPasadoDeGrogXD ]
```

“4b. Se sabe que algunos tripulantes son espías de la corona. Estos piratas se comportan igual que los piratas comunes a excepción de que:
● nunca están pasados de grogXD
● para animarse a saquear una víctima se tienen que dar las condiciones explicadas anteriormente y además tener el ítem permiso de la corona”




```
EspiaDeLaCorona >> estasPasadoDeGrogXD
	^false

EspiaDeLaCorona >> teAnimasASaquear: unaVictima
	^(super teAnimasASaquear:unaVictima)
		and: [ self tenes: 'permiso de la corona']
```

“5a) Saber cuántos tripulantes de un barco están pasados de grogXD”

```
Barco >> cantidadTripulantesPasados
	^self tripulantesPasados size

Barco >>  tripulantesPasados
	^tripulantes select:
		[ :unTripulante | unTripulante estasPasadoDeGrogXD ]
```

“5b) Saber cuántos tipos distintos de items se juntan entre los tripulantes de un barco que están pasados de grogXD (es decir, los ítems sin repetidos)”

```
Barco >> cantidadItemsDistintosEntreLosPasados
	| itemsJuntados |
	itemsJuntados = Set new.
	self tripulantesPasados do: [ :p |
		itemsJuntados addAll: p items
	].
	^itemsJuntados
```

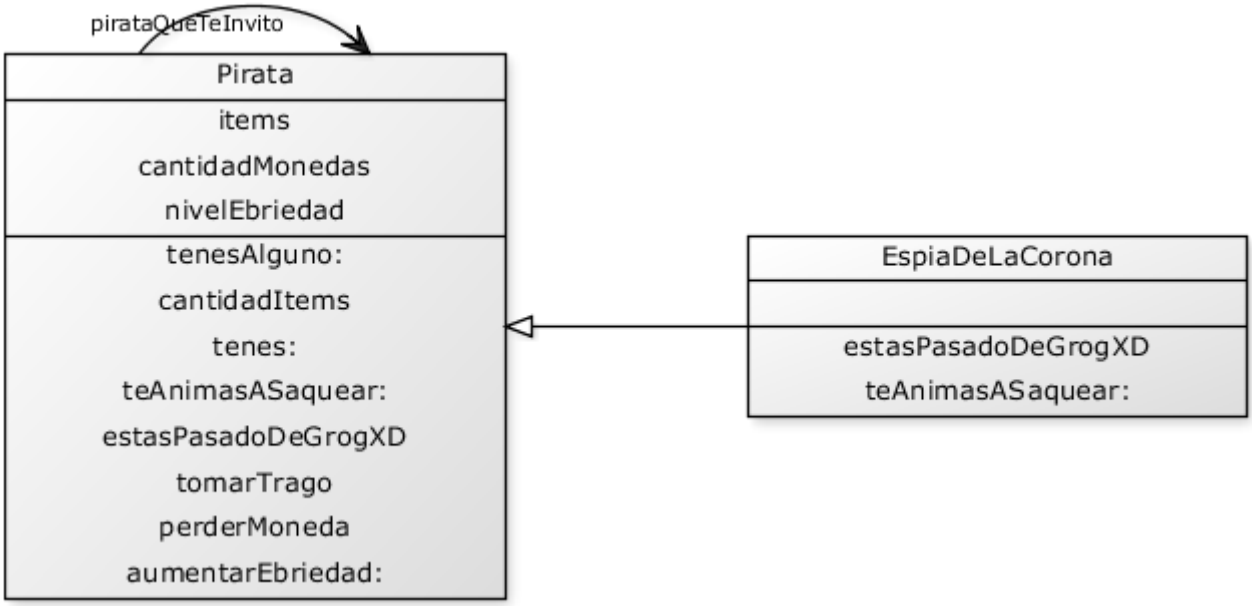
"O bien..."

```
^self tripulantesPasados
      inject: Set new
      into:[:itemsJuntados :p | itemsJuntados union: p items].
```

“5c) Obtener el tripulante pasado de grogXD con más dinero de la tripulación de un barco. ¡Gaaar! ¡Ojo con repetir código!”

```
Barco >> pirataMasPasado
      ^self tripulantesPasados asSortedCollection:
            [ :pirataQueVaAntes :pirataQueVaDespues |
                  pirataQueVaAntes nivelEbriedad
                        > pirataQueVaDespues nivelEbriedad ]
```

“6. Cada tripulante conoce qué tripulante del barco lo invito. Se quiere conocer quién es el tripulante de un barco pirata que invito (satisfactoriamente) a más gente”




```
Barco >> pirataMasInvitador
      ^tripulantes asSortedCollection:
            [ :pirataQueVaAntes :pirataQueVaDespues |
            (pirataQueVaAntes cantidadTripulantesInvitados: self)
                        >
            (pirataQueVaDespues cantidadTripulantesInvitados: self)]
```

“Parece razonable que un pirata nos sepa decir la cantidad de gente que invito en un barco”

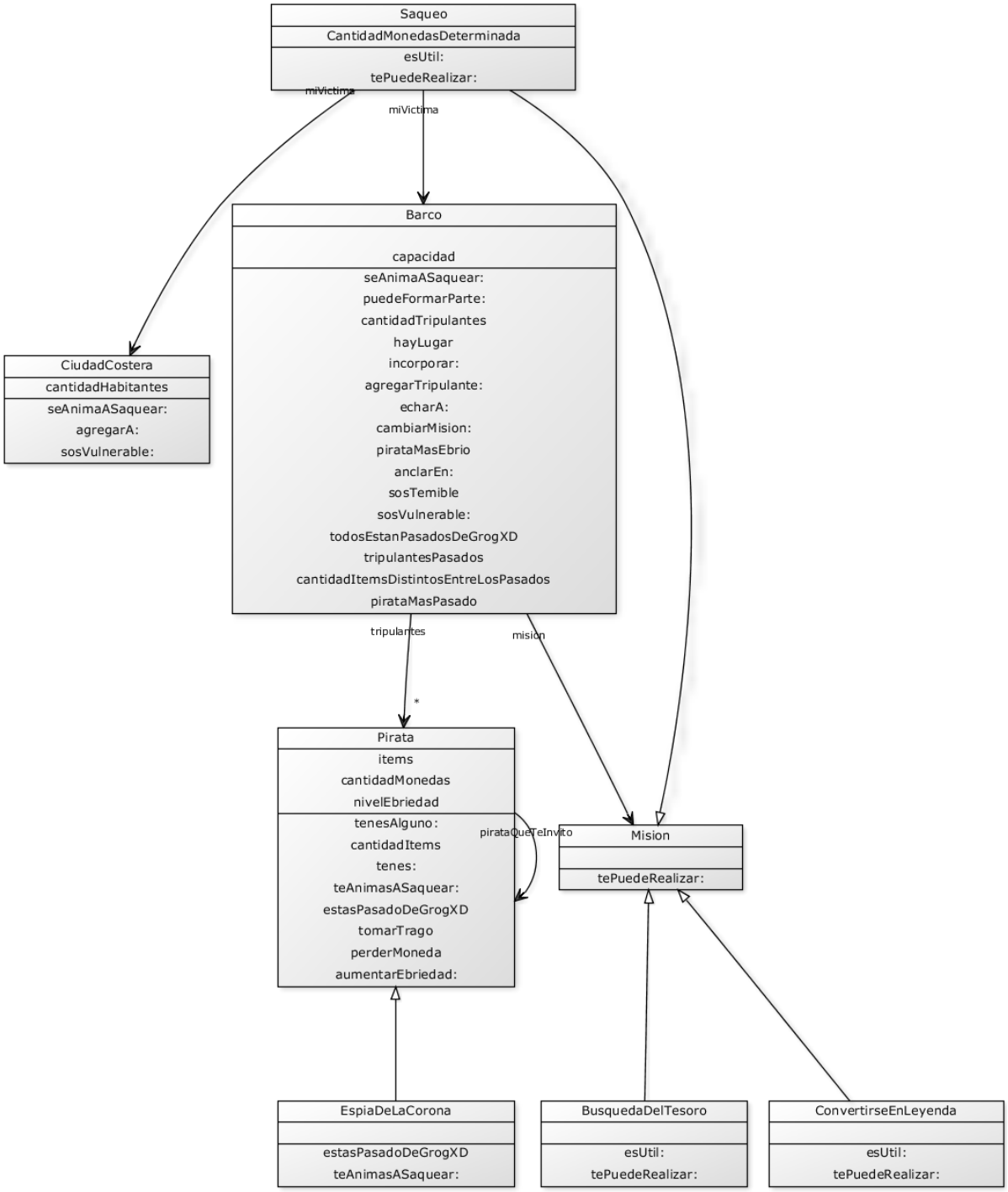
```
Pirata >> cantidadTripulantesInvitados: unBarco
```

“Pero un pirata no conoce a quien invito sino quién lo invito a él … parece que hay que delegar la responsabilidad en el barco”

```
Pirata >> cantidadTripulantesInvitados: unBarco
      ^unBarco cantidadTripulantesInvitadosPor: self

Barco >> cantidadTripulantesInvitadosPor: unPirata
      ^tripulantes count: [ :p | p pirataQueTeInvito = unPirata ]
```

"O bien"

```
      ^(tripulantes select:
            [ :p | p pirataQueTeInvito = unPirata ]) size
```



El diagrama se puede editar pegando lo siguiente en www.yuml.me

```
[Pirata|items;cantidadMonedas;nivelEbriedad|
tenesAlguno:;cantidadItems;tenes:;teAnimasASaquear:;estasPasadoDeGro
gXD;tomarTrago;perderMoneda;aumentarEbriedad:]
[Pirata]^[EspiaDeLaCorona||estasPasadoDeGrogXD;teAnimasASaquear:]
[Pirata]pirataQueTeInvito->[Pirata]
[Barco|;capacidad|
seAnimaASaquear:;puedeFormarParte:;cantidadTripulantes;hayLugar;inco
rporar:;agregarTripulante:;echarA:;cambiarMision:;pirataMasEbrio;anc
larEn:;sosTemible;sosVulnerable:;todosEstanPasadosDeGrogXD;tripulant
```

esPasados;cantidadItemsDistintosEntreLosPasados;pirataMasPasado]
[BusquedaDelTesoro||esUtil:;tePuedeRealizar:]
[ConvertirseEnLeyenda||esUtil:;tePuedeRealizar:]
[Saqueo|CantidadMonedasDeterminada|esUtil:;tePuedeRealizar:]
[Mision||tePuedeRealizar:]
[Mision]^[ConvertirseEnLeyenda]
[Mision]^[BusquedaDelTesoro]
[Mision]^[Saqueo]
[Barco]mision->[Mision]
[Barco]tripulantes->*[Pirata]
[Saqueo]miVictima->[Barco]
[CiudadCostera|cantidadHabitantes|
seAnimaASaquear:;agregarA:;sosVulnerable:]
[Saqueo]miVictima->[CiudadCostera]