Práctico de Programación 4 – clase 7

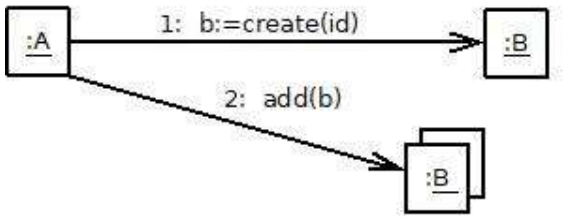
Práctico 4, Ejercicio 2:

Comentario general: una asociación entre una clase A y una clase B, en el modelo de dominio se representa mediante una línea que une las dos clases y tiene un nombre que indica el sentido de lectura. Por ejemplo, entre Casa e Inmobiliaria, la asociación puede ser "Inmobiliaria vende Casa" o "Casa vendida por Inmobiliaria". La semántica es la misma, la única diferencia es el sentido de lectura. En un modelo de dominio (que es una representación de conceptos de la realidad, aún NO son clases de software) se asume que es posible navegar hacia ambos lados (una casa ve a sus inmobiliarias y una inmobiliaria ve a sus casas). En la etapa de diseño debe decidirse el sentido de las navegabilidades. Para el caso de este ejercicio, debe decidirse si:

- A mantiene una colección de referencias a B (porque la multiplicidad es 0..*).
- B mantiene referencia a su único A (porque la multiplicidad es 1).
- Las dos opciones anteriores a la misma vez.

Desde el punto de vista de diseño, la selección de una opción entre las tres anteriores dependerá de consideraciones acerca de la facilidad de implementación de las operaciones, la eficiencia de las operaciones y la dificultad de mantener las estructuras de referencias entre objetos. Notar que siempre la tercera opción será la más flexible, sin embargo, es la más compleja en términos de implementación. En este ejercicio se pide modelar las interacciones entre objetos, según diferentes opciones de diseño.

a) Si el link es unidireccional de A hacia B, significa que la instancia de clase A debe guardar la referencia a la instancia de clase B recién creada.

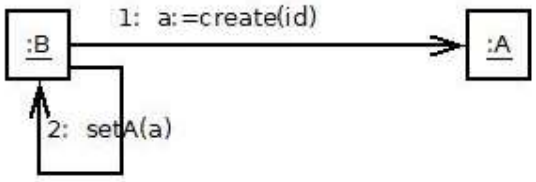


Si bien en la etapa de diseño aún no se piensa en la implementación, a continuación vemos cómo sería el código C++ de las clases, para ejemplificar una posible implementación de este diseño.

```
class A {
    private:
        int idA;
        Coleccion misB;
}
```

```
class B {
    private:
        int idB;
}
```

b) Si el link es unidireccional de B hacia A, significa que la instancia de B debe guardar la referencia a la instancia de clase A recién creada.

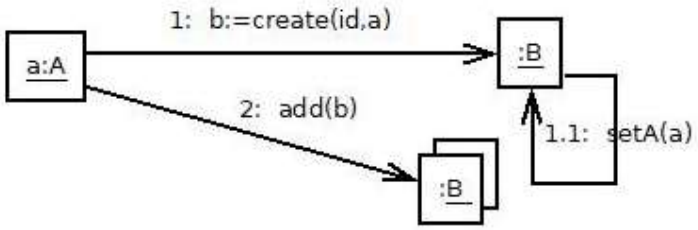



```
class A {
    private:
        int idA;
}
```

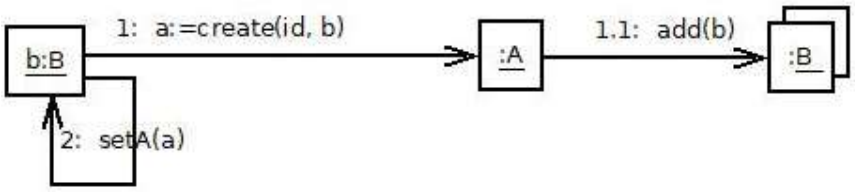
```
class B {
    private:
        int idB;
        A *miA;
    public:
        void setA(A *);
}
```

c)

Para el diagrama de la parte a) notar que ahora la instancia a:A (a de clase A) se pasa como parámetro a sí misma en la operación create, de modo que la instancia de clase B pueda guardar una referencia a ella (operación setA).



Para el diagrama de la parte b) notar que ahora la instancia b:B (b de clase B) se pasa como parámetro a sí misma en la operación create, de modo que la instancia de clase A pueda guardar una referencia a ella (operación add, dado que es una colección de muchas instancias de clase B).



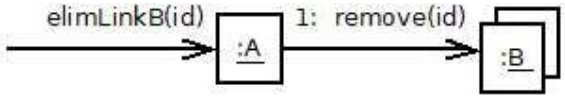
El código C++ para estos dos últimos diagramas, es una combinación de los códigos vistos más arriba, por lo tanto es una implementación más compleja:

```
class A {
    private:
        int idA;
        Coleccion misB;
}
```

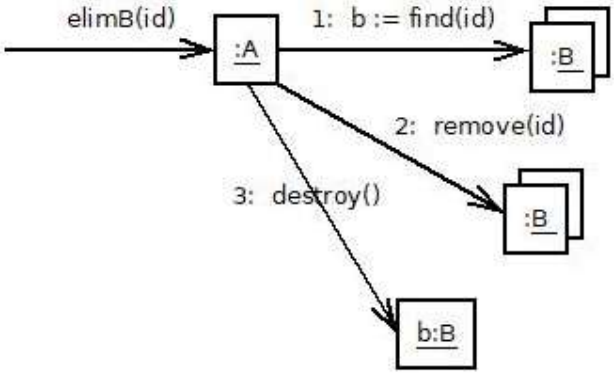
```
class B {
    private:
        int idB;
        A *miA;
    public:
        void setA(A *);
}
```

d) En términos generales (aunque no siempre estrictamente), si una clase A es responsable de crear instancias de una clase B, también es responsable de destruirlas. La destrucción de links NO es lo mismo que la destrucción de instancias. Destruir un link implica "desengancharse" de un objeto, pero no eliminarlo.

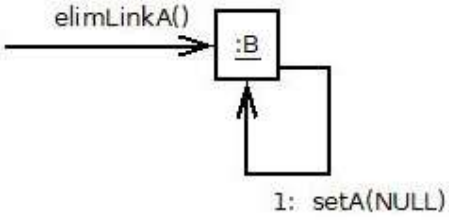
Si queremos eliminar el link a la instancia de B que creó A en el diagrama de la parte a) de este ejercicio, sería como sigue:



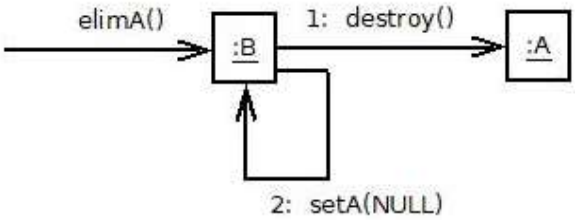
Si además queremos eliminar la instancia, el diagrama sería:



Si queremos eliminar el link a la instancia de A que creó B en el diagrama de la parte b) de este ejercicio, sería como sigue:



Si además queremos eliminar la instancia, el diagrama sería:



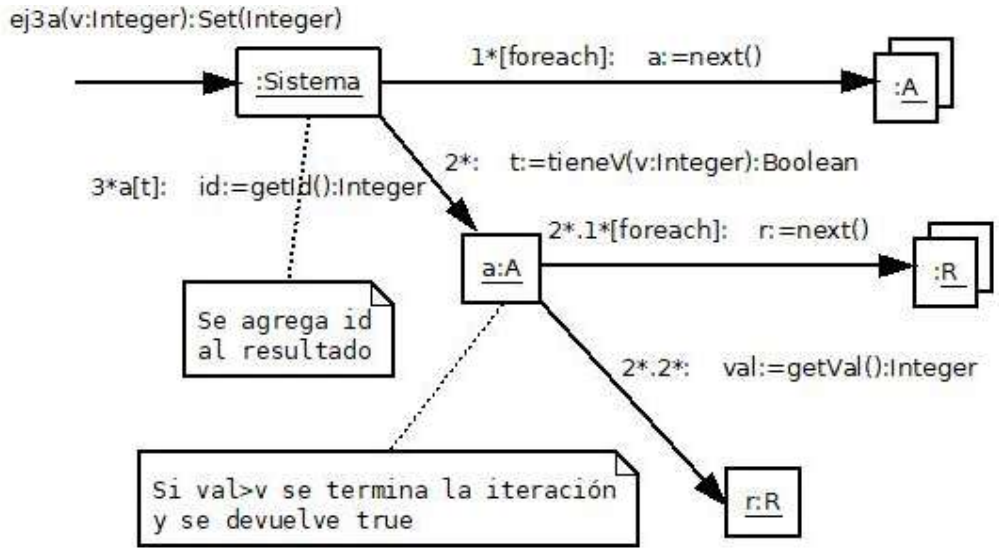
Práctico 4, Ejercicio 3:

El objetivo de este ejercicio es ver cómo una operación puede ser más compleja, dependiendo de las decisiones de diseño que se hayan tomado respecto al sentido de las navegabilidades de las asociaciones.

a) Si R es navegable de A hacia B, significa que la asociación se diseñó de la siguiente forma:
   - Cada instancia de A tiene referencia a muchas de R (por la multiplicidad * del lado de B).
   - Cada instancia de R tiene referencia a una (y solo una) instancia de B (por definición de asociación). Notar que el par (a1,b1) puede estar incluido una sola vez en la asociación $R \subseteq A \times B$.

   El esquema general de la búsqueda para resolver el ejercicio es:

   - Recorrer todas las instancias de clase A en el sistema.
   - A cada una de esas instancias de A preguntarle si está asociada con algún B y si además en el correspondiente tipo asociativo R tiene un valor mayor que v.
   - Si se cumple lo anterior (condición t), agregar el id de la instancia de A al resultado.



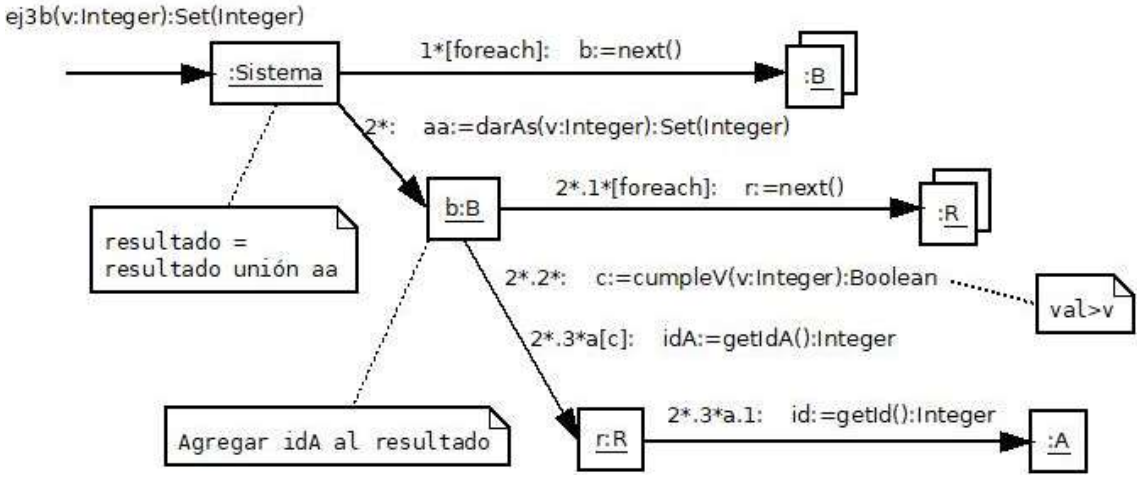
b) Si R es navegable de B hacia A, significa que la asociación se diseñó de la siguiente forma:
   - Cada instancia de B tiene referencia a muchas de R (por la multiplicidad * del lado de A).
   - Cada instancia de R tiene referencia a una (y solo una) instancia de A (por definición de asociación).

   Notar que la estructura es simétrica a la de la parte a) del ejercicio. Sin embargo, el diseño de la operación no es simétrico, resultando más complejo.

   El esquema general de la búsqueda ahora es:

   - Recorrer todas las instancias de clase B en el sistema.

- A cada una de esas instancias de B pedirle los id de todas las instancias de A con las que está asociada y que además tienen un valor mayor que v en el correspondiente tipo asociativo R.
- Agregar el conjunto de esos id de A al resultado, mediante la operación unión de conjuntos, para eliminar repetidos. Notar que una misma instancia de A puede aparecer varias veces en el resultado aa a través de la iteración sobre las diferentes instancias de B.



Práctico 4, Ejercicio 8:

Los objetivos de este ejercicio son:

- Ejercitar la aplicación de criterios de asignación de responsabilidades.
- Discutir el impacto de algunas decisiones de diseño acerca del aspecto dinámico del sistema, sobre la estructura estática del mismo.

a)

Un posible esquema general de la búsqueda para generar el resultado especificado en el contrato de la operación `darGanadores` es el siguiente:

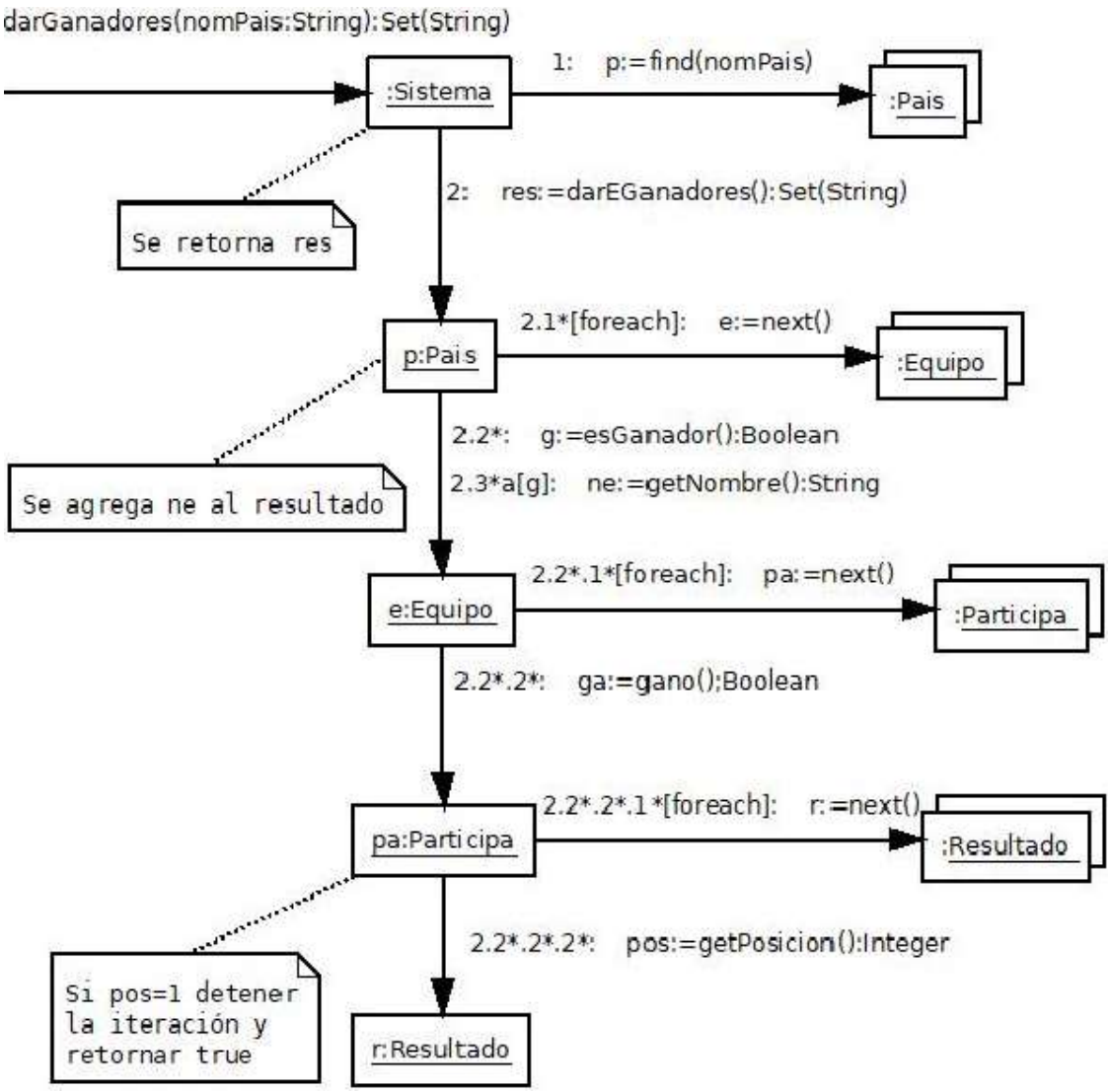
- Buscar el país.
- Pedir a ese país que devuelva la lista de sus equipos ganadores. Para eso, a cada equipo se le preguntará si ganó alguna vez. Para determinar esto, el equipo debe recorrer todos los torneos donde participó y para cada uno ver si ganó en algún año.

Respecto a la asignación de responsabilidades, notar que se aplica reiteradamente el criterio GRASP *experto* (ver teórico, clase 11 - Diseño: GRASP), de la siguiente forma:

- Una vez que se encuentra al país identificado por el nombre indicado por el parámetro, se le delega la responsabilidad de obtener sus equipos ganadores.
- El país delega al equipo la responsabilidad de saber si fue ganador.
- Equipo delega en participa y así hasta llegar a los resultados.

Notar que de esta forma se logra un diseño que además cumple con otros criterios GRASP como ser *alta cohesión y bajo acoplamiento*. Esto se ve reflejado en un diagrama donde los mensajes se

propagan en profundidad y cada instancia no envía muchos mensajes. Si por el contrario, por ejemplo, el país fuera el encargado de interactuar con participa o incluso con resultado, se estaría aumentando el acoplamiento entre clases; también se estaría violando el criterio de *no hables con extraños*. Esto se reflejaría en un diagrama con más amplitud que profundidad, donde un objeto envía muchos mensajes a otros objetos de diferentes clases, lo que no es deseable.



b)

Un posible esquema general de la búsqueda para generar el resultado especificado en el contrato de la operación `darTorneosInter` es el siguiente (se omite el diagrama):

- Recorrer todos los torneos.
- A cada uno preguntarle si es internacional. Para eso, el torneo deberá recorrer sus participa (cada uno tiene un equipo) y comparar con los demás para determinar si hay equipos de diferentes países (se necesita acceder a país desde equipo).

Notar que la resolución de esta operación queda sencilla si se navegan las asociaciones en el sentido contrario en que se navegaron para resolver la parte a) del ejercicio. Por lo tanto, en este caso es

posible que lo más conveniente sea mantener asociaciones bidireccionales en los casos que se necesite (ver discusión al respecto en los ejercicios anteriores).

Observación final (sobre todos los ejercicios):

- Todos los mensajes enviados a multiobjetos (que representan colecciones) son exclusivamente operaciones de colecciones (next, add, find, remove, etc), ver teórico de clase 10 (Diseño: Diagramas de Comunicación).
- Debido al software utilizado para realizar los diagramas en este documento, hay una desviación con respecto a la sintaxis de UML. Cuando un objeto envía un mensaje a otro, la línea que los une no lleva flecha. La flecha corresponde al mensaje y se dibuja debajo de su nombre.