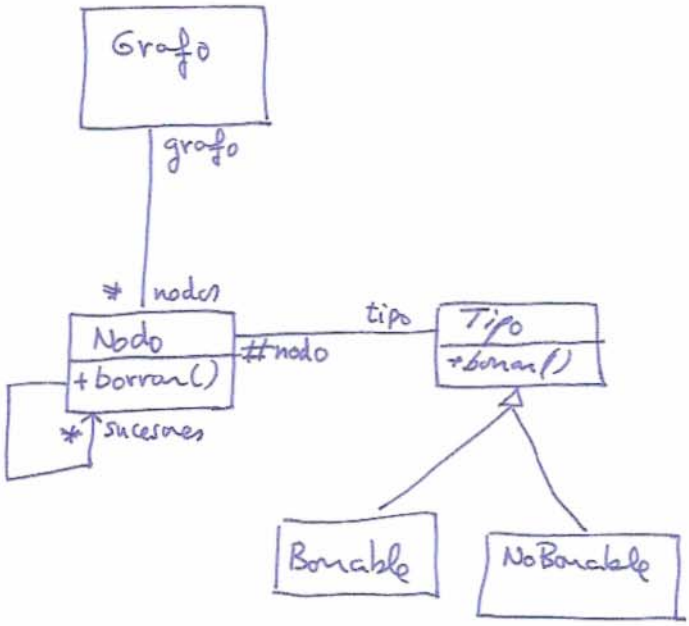
Ingeniería del Software II — Tercer examen corto. Modelo 1. Alumno: _____

1) Un determinado tipo de grafo puede tener nodos borrables y no borrables. Cuando sobre un nodo borrable *n* se ejecuta la operación *borrar()*, ocurren dos cosas: (1) el nodo *n* desaparece del grafo y (2) la operación *borrar()* se transmite a los nodos que sean alcanzables desde *n*. Si la operación *borrar()* se ejecuta sobre un nodo no borrable, entonces no pasa nada.

(a) Represente con un diagrama de clases este tipo de grafo, utilizando el patrón *Estado*, y sabiendo que no puede utilizar la clase *Arco*. **3 puntos**

(b) Muestre el código o pseudocódigo de la operación *borrar()* en los diferentes tipos de su diagrama. **2 puntos**

(a) Un grafo tiene muchos nodos.
Cada nodo tiene una lista de adyacentes o sucesores, que son nodos.
El tipo del nodo lo delegamos a una clase Tipo, que representa el "estado" del patrón Estado.



(b) En el tipo Nodo, la operación "borrar()" no hace nada: sólo le dice a su estado que borre. El estado, en función de que sea Borrable o NoBorrable, hace una cosa u otra.

```
Nodo::borrar() {
    this.tipo.borrar();
}
```

```
Borrable::borrar() {
    this.nodo.grafo.eliminar(this.nodo);
    for (Nodo n: this.nodo.sucesores)
        n.borrar();
}
```

```
NoBorrable::borrar() {
    return;
}
```

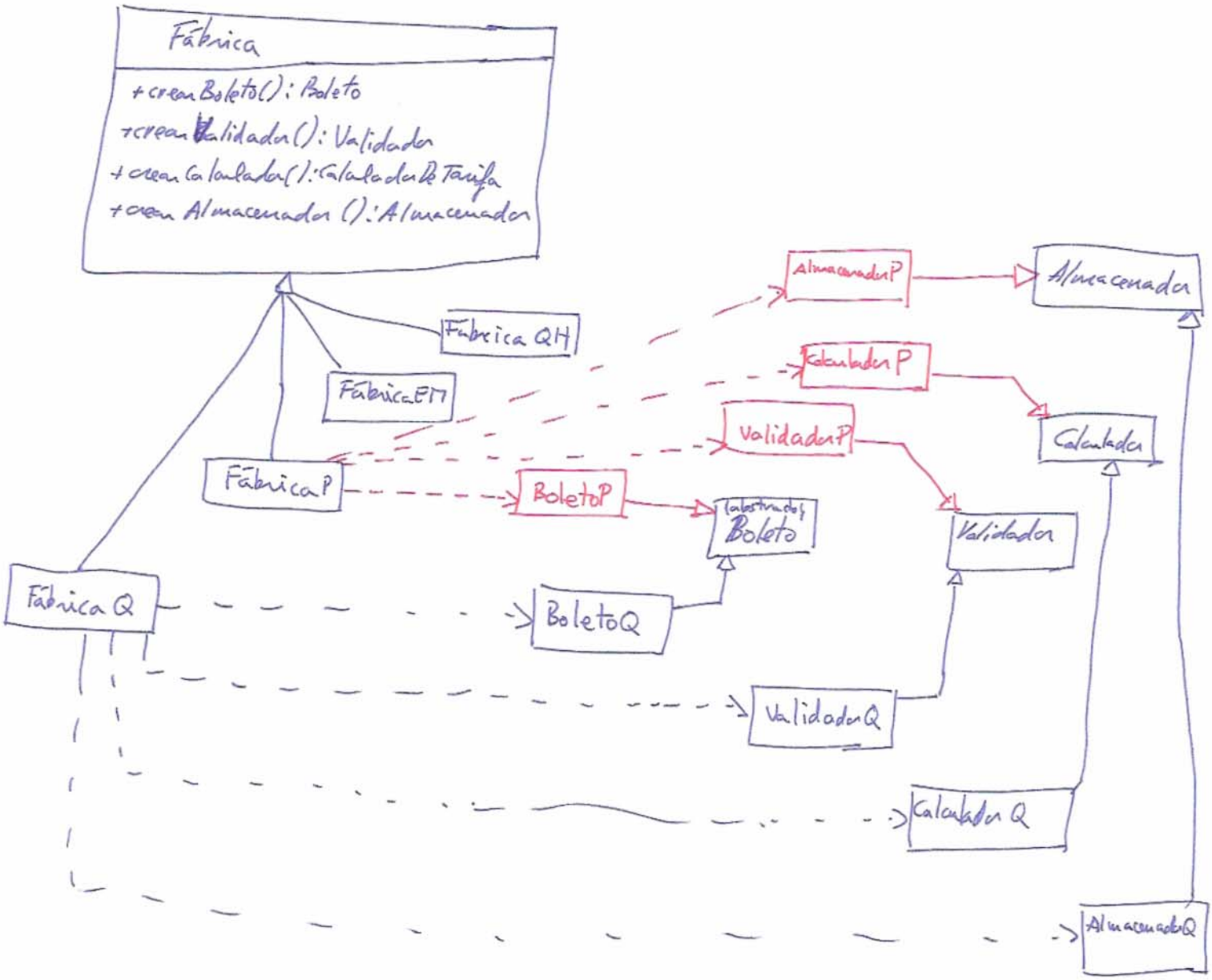
Ingeniería del Software II — Tercer examen corto. Modelo 1. Alumno: _____

2) Se dispone de un sistema que recibe por un socket ristras de bytes que representan diferentes tipos de boletos de varios juegos de azar (quinielas, primitivas, euromillones, quinigoles, etc.). Cada vez que llega una ristra de bytes, el sistema debe instanciar:

- El subtipo adecuado de *Boleto* (como en su práctica)
- El subtipo adecuado de *Validador* (que tiene una operación abstracta *esValido(Boleto):boolean)*
- El subtipo adecuado de *CalculadorDeTarifa* (que tiene una operación abstracta *tarifa(Boleto):double)*
- El subtipo adecuado de *Almacenador* (que tiene una operación abstracta *insert(Boleto)*, que inserta el boleto en la tabla correspondiente)

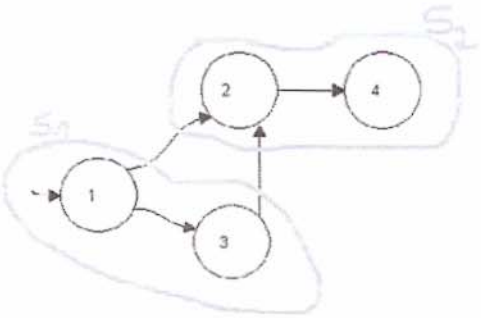
Represente la estructura de este sistema mediante el patrón *Fábrica Abstracta.* **2 puntos**

Tenemos tantos lotes como tipos de juegos, y necesitamos una fábrica concreta por cada lote.



Ingeniería del Software II — Tercer examen corto. Modelo 2. Alumno: _____

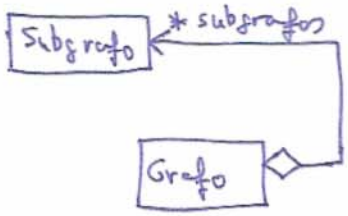
1) Se puede considerar que un grafo dirigido está formado por subgrafos dirigidos, ocurriendo que estos subgrafos están unidos por arcos. Así, el grafo de la figura de abajo está formado por los subgrafos S_1 y S_2, que están unidos por los arcos 1-2 y 3-2. El grafo de la figura podría igualmente considerarse un subgrafo, de manera que, unido a otro subgrafo, podría formar un grafo más grande.



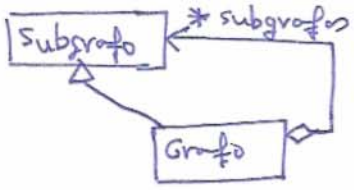
(a) Represente con un diagrama de clases este tipo de grafo, utilizando los patrones *Composite* y/o *Chain of responsibility*, sabiendo que no puede utilizar la clase *Arco*. **3 puntos**

(b) Se desea dotar al grafo que Vd. acaba de modelar de una operación *getNodos():int*, que devuelve el número de nodos del grafo. Muestre el código o pseudocódigo de la operación u operaciones necesarias para que se calcule ese valor. **2 puntos**

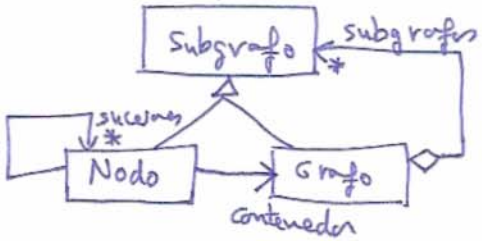
(a) Nos dicen que "un grafo está formado por subgrafos". Veamos:



Igualmente, se dice: "El grafo de la figura podría considerarse un subgrafo":



Podemos considerar que un Nodo es un subgrafo, con una lista de sucesores. Además, cada nodo pertenece a un grafo.



(b) En Subgrafo, getNodos es abstracta.

En Nodo:
```
int getNodos() {
    return 1;
}
```

En Grafo:
```
int getNodos() {
    int r = 0;
    for (Subgrafo s: this.subgrafos)
```

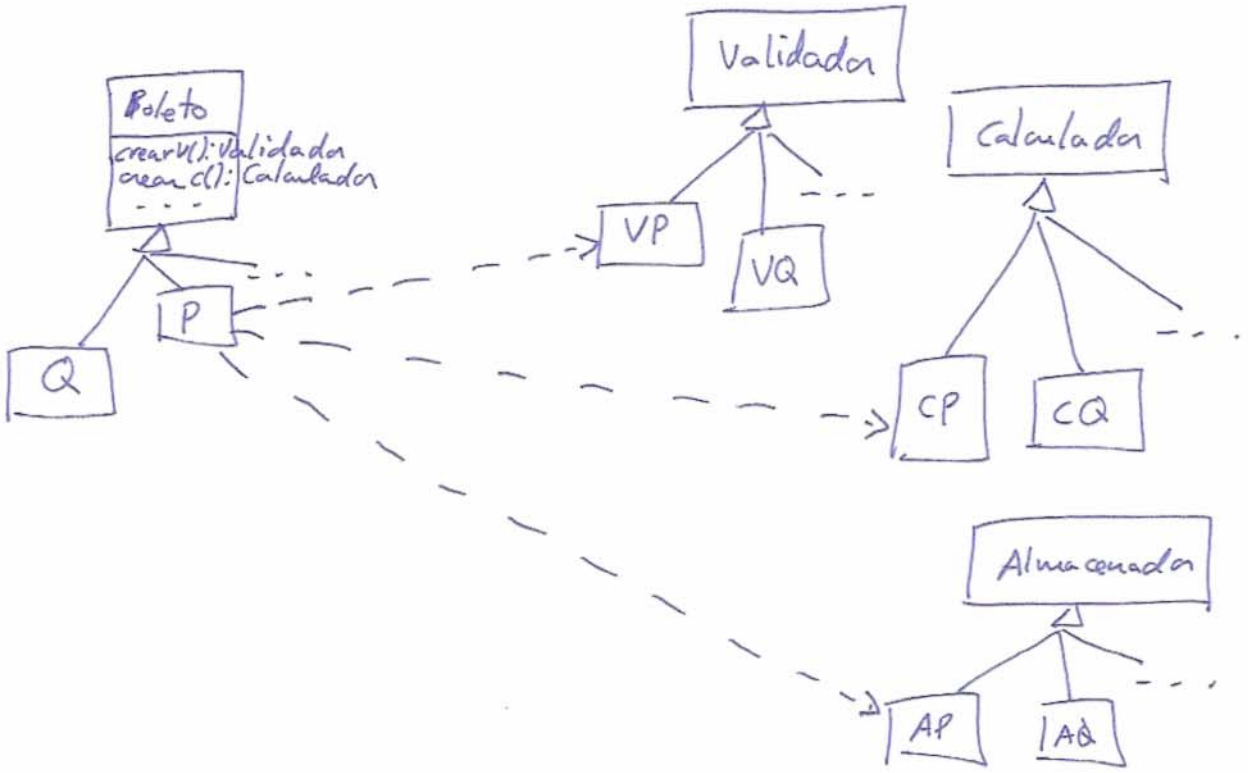
Ingeniería del Software II — Tercer examen corto. Modelo 2. Alumno: _____

2) Se dispone de un sistema que recibe por un socket ristras de bytes que representan diferentes tipos de boletos de varios juegos de azar (quinielas, primitivas, euromillones, quinigoles, etc.). Cada vez que llega una ristra de bytes, el sistema debe instanciar:

- El subtipo adecuado de *Boleto* (como en su práctica)
- El subtipo adecuado de *Validador* (que tiene una operación abstracta *esValido(Boleto):boolean)*
- El subtipo adecuado de *CalculadorDeTarifa* (que tiene una operación abstracta *tarifa(Boleto):double)*
- El subtipo adecuado de *Almacenador* (que tiene una operación abstracta *insert(Boleto)*, que inserta el boleto en la tabla correspondiente)

Represente la estructura de este sistema mediante el patrón *Builder*. **2 puntos**

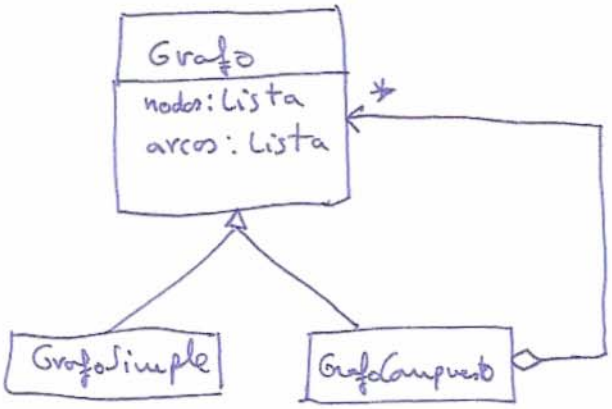
Hacemos que el Builder cree, por ejemplo, el Boleto:



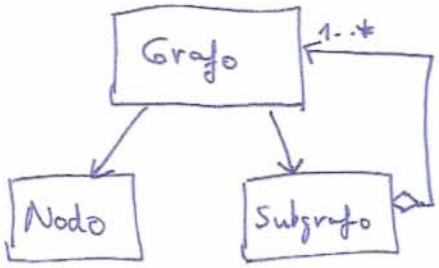
Modelo 2.

Ejercicio 1:

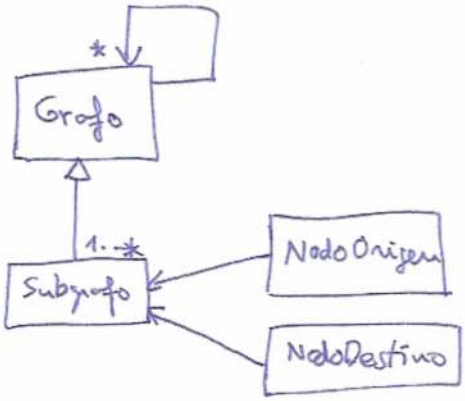
Sería incorrecto representar el grafo de las siguientes formas:



← ¿Qué son "nodos" y "arcos"? Son listas, pero, ¿de qué tipo de objeto? Conviene representar las colecciones como asociaciones en las q. se enlazan los dos tipos relacionados.



← Esta solución se ha dado en algún examen. Al no haber multiplicidades, se está diciendo que todo grafo tiene un nodo y un subgrafo. Quizás se haya confundido la asociación (↓) con la herencia (△). En esta solución no hay patrón compuesto ni chain of responsibility.



← Esta solución tiene errores muy serios:

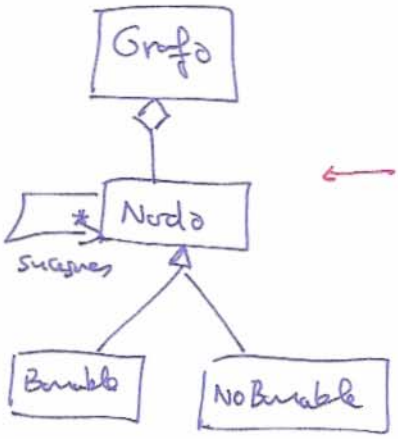
1º) Se está poniendo multiplicidad en la relación de herencia.

2º) Se crean dos tipos (NodoOrigen y NodoDestino) para representar el hecho de que un nodo puede ser origen y destino.

Modelo 1.

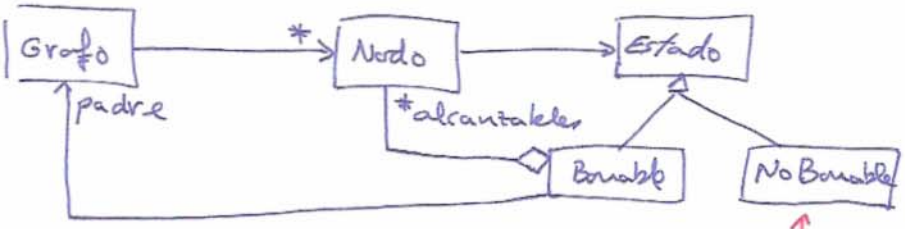
Ejercicio 1.-

Con el Estado se delega el comportamiento a una clase asociada. Así pues, no es correcta la siguiente solución:



← Además, se ha dividido la multiplicidad.

En esta solución, se delega el tipo del nodo a un Estado, pero luego se crean relaciones incorrectas:



Los nodos no bonables también tienen sucesores y están contenidos en un grafo padre.

En esta otra, el autor se olvida del tipo Nodo, aunque lo menciona en el compartimento de campos del Grafo: