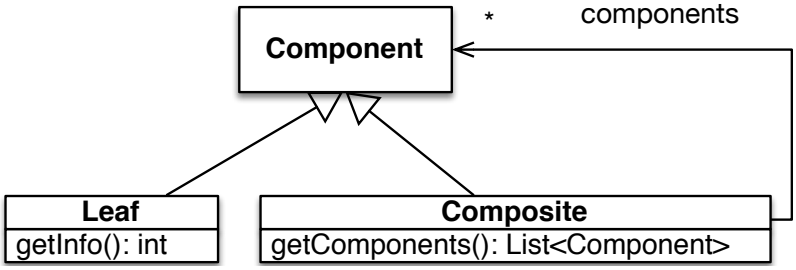
Problema 1. (2 punts)

- Què és un `ResultSet` en JDBC? Què permet fer?
- Quin inconvenient té l'estratègia d'una taula per jerarquia en JPA ?

Problema 2. (5 punts)

Donat el següent disseny:



que utilitza el **patró composite** en un cas en el que els únics nodes amb informació, en aquest cas un enter, són les fulles.
El que es vol és:

a) Afegiu el que calgui per tal d'aplicar el patró visitor per a poder afegir operacions sobre els components.
   Tingueu en compte la següent consideració:
   - El recorregut dels components d'un composite el realitza cada visitant concret al seu mètode `visit`
   - El mètode `getComponents` retorna una llista no modificable (`Collections.unmodifiableList`)

b) Implementeu dos visitants concrets:
   - `CountVisitor`: que compta el nombre de fulles
   - `SumVisitor`: que suma la info de les fulles

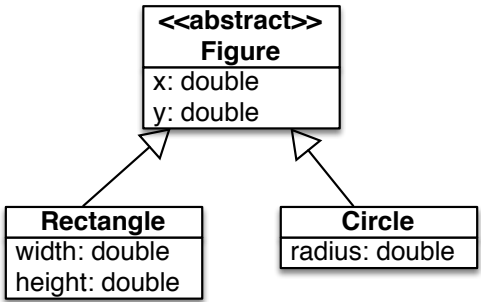
   Tingueu en compte la següent consideració:
   - No està garantit que els components formen un arbre, de manera que haurem de protegir-nos contra el problema de caure en un bucle a l'hora de fer les visites.

c) Afegiu la infraestructura necessària per a simplificar la implementació de visitants que tinguin la mateixa estructura que els dos anteriors. Quin patró fareu servir i per què?

d) Mostreu que la implementació es simplifica mostrant el `SumVisitor` que usa la infraestructura creada a l'apartat anterior.

Problema 3. (3 punts)

Donada el següent diagrama de classes:



en el que totes les classes que apareixen són **immutables** es vol crear un mecanisme de *creació fluïda* d'instàncies.

Concretament es vol poder crear `Figures` de la següent manera:

```
1 Rectangle rect = (Rectangle) Figure
2                     .create()
3                     .at(x, y)
4                     .withDimensions(width, height)
5                     .execute();
6
7 Circle circle = (Circle) Figure
8                       .create()
9                       .at(x, y)
10                      .withRadius(radius)
11                      .execute();
```

a) Quin patró creus que és aplicable en aquest cas i perquè?
b) Implementeu la vostra solució tenint en compte les següents restriccions:
   - Com a mínim ha d'haver un `at` i, en cas d'haver-hi varis, es tindrà en compte el **primer** d'ells.
   - No poden haver `withDimensions` i `withRadius` alhora.
   - En cas de `withDimensions` o `withRadius` repetits, es tindrà en compte el **darrer** d'ells.
   - Els `at` i els `withXXXX` poden venir en **qualsevol ordre**.
   - Com a mínim ha d'haver un `withDimensions` o un `withRadius`.

   Si alguna de les restriccions no es compleix, es llençarà l'excepció no comprovada predefinida `IllegalStateException`.