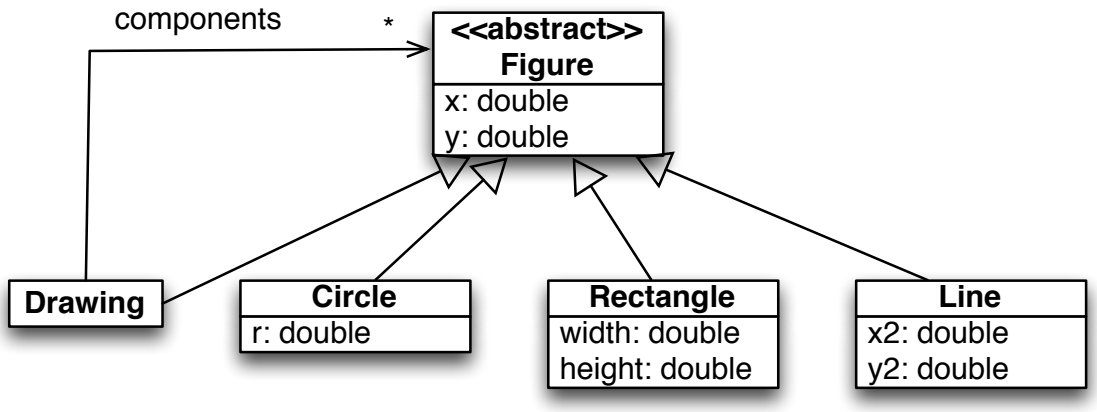
Problema 1. (5 puntos)

Dada la jerarquía de clases:



En la que, además de constructores que reciben valores para todas las variables de instancia, hay getters y setters para todas las propiedades.

Se desea añadir la posibilidad de tener observadores de las mismas, de manera que si alguno de los atributos se modifica, los objetos de la capa presentación interesados son notificados.

Se pide que:

- Usando la infraestructura de Java, haced las modificaciones necesarias sobre la clase Figure, para poder observar sus instancias.
- Implementad la clase `Figure` y `Circle` completamente.
- Suponed ahora que `Figure` no pudiera definirse como subclase de otra. ¿Cómo solucionaríais ahora el problema? Mostrad la nueva implementación de la clase `Figure`.

Problema 2. (5 puntos)

Considerad la jerarquía de clases del problema anterior, pero suponed ahora que todas las clases simples son **inmutables** (es decir, todas sus variables de instancia son **final**).

Lo que querremos es definid un mecanismo de construcción flexible de figuras simples tal que, por ejemplo permita hacer:

```
Circle    c=(Circle) Figure.create().at(10,20).withRadius(40).do();
Rectangle r=(Rectangle)Figure.create().at(20,30).withDims(30,30).do();
Line      l=(Line)Figure.create().from(10,20).to(-10,30).do();
```

Si falta información para hacer una figura o ésta es contradictoria (p.e. una figura con radio y dimensiones) se devuelve **null**.

¿Qué patrón se está usando?

Implementad la nueva funcionalidad pedida. No hace falta que en las clases simples implementéis constructores y getters.

NOTA: **No utilicéis boxing/unboxing automático entre Double y double**.