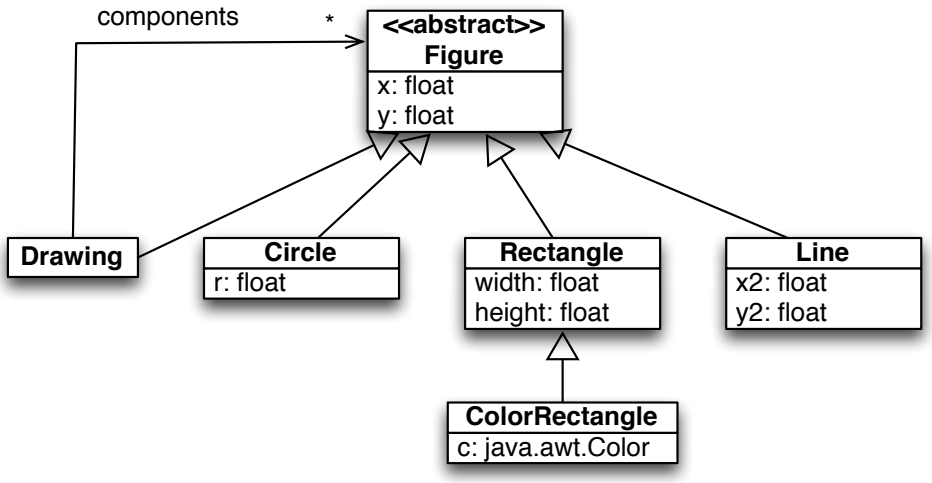
Nombre: _____ DNI: _____

Problema 1. (2 puntos)

Describe brevemente (máximo 5 líneas por patrón) la intención de cada uno de los dos patrones que habéis trabajado en vuestro grupo.

Problema 2. (5 puntos)

Dada la jerarquía de clases:



En la que hay operaciones para modificar las propiedades de las figuras, se desea añadir la posibilidad de tener observadores de las misma, de manera que si alguno de los atributos que afectan la visualización de las mismas 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 (2 puntos)
- Mostrad la nueva implementación del método setRadius de la clase Circle (1 punto)
- Suponed ahora que Figure no pudiera definirse como subclase de otra clase. ¿Cómo solucionaríais ahora el problema? Mostrad la nueva implementación de la clase Figure (2 puntos).

Problema 3. (3 puntos)

Considerad la jerarquía de clases del problema anterior, pero suponed ahora que todas las clases son **inmutables**. Lo que querremos es definid un mecanismo de construcción flexible de figuras simples tal que, por ejemplo permita hacer:

```
c = Figure.create().at(10,20).withRadius(40).do();
```

Y c es una instancia de Circle. Pero si hacemos:

```
r = Figure.create().at(20,30).withDims(30,30).do();
```

Y ahora r es una instancia de Rectangle. Para las Lines:

```
l = Figure.create().from(10,20).to(-10,30).do();
```

Y para ColorRectangle tendremos:

```
rc = Figure.create().at(20,30).withDims(30,30).in(Color.RED).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 funcionalidad pedida.