CLASES INTERNAS

Como su nombre indica, una clase interna, es una clase que se encuentra dentro de otra, de la siguiente forma:

```java
public class ClasesInternas {

    class Interior {

    }

}
```

¿Por qué nos interesa crear clases internas?:

- Acceder a campos privados de una clase, desde otra, sin crear *getters*.
- Ocultar una clase de otras pertenecientes al mismo paquete.
- Crear clases “anónimas”, muy útiles para gestionar eventos o retrollamadas.
- Cuando solo una clase debe acceder a otra clase.

Voy a poner un ejemplo personal, imaginemos que queremos que una función nos devuelva dos tipos de datos, en vez solamente uno, podemos crear un objeto que contenga estos dos tipos como atributos, y devolver dicho objeto, de la siguiente forma:

```java
public class ProblemaAjedrez {

    private static class Solucion {

        // atributos
        private boolean resuelto;
        private char[][] tablero;

        // constructor
        public Solucion (boolean resuelto, char[][] tablero) {

            this.resuelto = resuelto;
            this.tablero = tablero;

        }

    }

    public static Solucion miFuncion () {

        return new Solucion (true, new char[0][0]);

    }

}
```

Como veis la clase es *private* ya que no la puede usar otra clase, y además es ***static***, ya que de ésta forma decimos que la clase **Solucion** está anidada.

Lo que hemos hecho arriba, se podría haber realizado de la siguiente forma:

```java
public class ProblemaAjedrez {

    public static Solucion miFuncion () {

        return new Solucion (true, new char[0][0]);

    }

}

class Solucion {

    // atributos
    private boolean resuelto;
    private char[][] tablero;

    // constructor
    public Solucion (boolean resuelto, char[][] tablero) {

        this.resuelto = resuelto;
        this.tablero = tablero;

    }
}
```

Pero de ésta forma, otras clases que no sean **ProblemaAjedrez.java** pueden acceder a la clase **Solucion.java**, y nosotros no buscamos eso.

RECORDAD: Todas las clases anidadas, llevan *private* y *static*.

También podemos crear clases anidadas dentro de una función:

```java
public class ProblemaAjedrez {

    public static boolean miFuncion () {

        class Solucion {

            // atributos
            private boolean resuelto;
            private char[][] tablero;

            // constructor
            public Solucion (boolean resuelto, char[][] tablero) {

                this.resuelto = resuelto;
                this.tablero = tablero;

            }

        }

        return new Solucion (true, new char[0][0]).resuelto;

    }
}
```

Evidentemente no podemos devolver como dato en *miFuncion()* el objeto **Solucion**, ya que no existe en la clase **ProblemaAjedrez.java** pero, aun así, esto es solo un ejemplo:

La clase no puede llevar ningún modificar fuera de los habituales (*abstract, final*), por lo demás funciona de la misma forma, como veis he hecho que *miFuncion()* devuelva un booleano, y en return, hago que me devuelva el atributo ‘resuelto’ del Objeto **Solucion**, en el caso de aquí es una estupidez, pero es solo para que veáis el funcionamiento.