Clases Internas

Clases internas

Como su nombre indica una clase interna **es una clase que se encuentra dentro de otra clase.** Cuando nos encontramos con una clase interna, podemos decir que por un lado tenemos a la propia clase interna, y por otro a una clase externa contenedora. Pero **¿para qué son necesarias este tipo de clases?** A continuación, se exponen los principales motivos:

- **Para acceder a los campos privados de una clase desde otra clase**. En ocasiones necesitamos "romper" la encapsulación de los campos de una clase determinada para utilizar esos campos en otra clase diferente. Las clases internas en estos casos son una solución.
- **Para ocultar una clase de otra que pertenezca al mismo paquete.** Las clases internas son un excelente recurso para encapsular clases que por diferentes motivos necesitamos que permanezcan invisibles para el resto de clases del paquete.
- **Para crear clases internas anónimas.** Las clases internas anónimas ahorran mucho código al programar sobre todo a la hora de gestionar eventos y retro llamadas.
- **Cuando solo una clase debe acceder a los campos de ejemplar de otra clase**. En estos casos crearemos clases internas locales como trataremos más adelante.

En definitiva, una clase interna se crea cuando deseamos que haya una estrecha relación entre 2 clases cuyos métodos generalmente están relacionadas unos con otros. Cuando creamos una clase interna, las variables y métodos de la clase contenedora estarán disponibles para la clase interna, **incluso aquellos marcados como privados(private).**

Las clases internas también cuentan con limitaciones:

- Los nombres de los campos de la clase contenedora han de ser diferentes a los nombres de los campos de la clase interna.
- Las clases internas no pueden contener ningún miembro estático.
- No es posible crear un objeto de la clase interna sin tener un objeto de la clase externa contenedora.

```
class Uno{

    Dos obj2=new Dos();

    public void llamaImprime() {

        obj2.imprimeNombre();
    }

     class Dos{

        public void imprimeNombre() {

            System.out.println(nombre);
        }

    }

     private String nombre="Juan";

}
```

En este ejemplo la clase "Dos" es la clase interna mientras que la clase "Uno" es la clase externa o contenedora

Clases internas locales

Las clases internas locales son clases dentro de un método.

- **¿Cuándo se utilizan?** Cuando solo se va a utilizar (instanciar) la clase interna una vez.
- **¿Cuál es su objetivo?** Simplificar aún más el código. Su ámbito queda restringido al método donde son declaradas.

Ni siquiera la clase contenedora puede acceder a estas clases. Tan solo el método donde están definidas podrá acceder a los campos y métodos de estas clases. Son la antesala de las clases internas anónimas y lo que se persigue es simplificar el código al máximo.

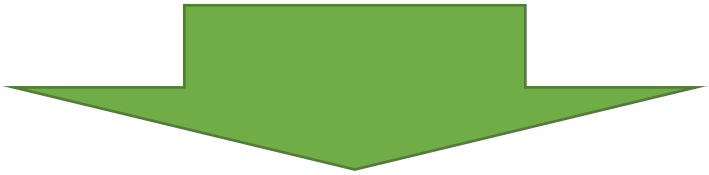
```
class Reloj{

    public void ejecutarTemporizador(int intervalo, boolean sonido){

        class DameLaHora implements ActionListener{

            @Override
            public void actionPerformed(ActionEvent e) {
                // TODO Auto-generated method stub

                Date ahora=new Date();

                System.out.println("Te pongo la hora cada 3 segundos: " + ahora);

                if (sonido) Toolkit.getDefaultToolkit().beep();

            }

        }

        ActionListener oyente=new DameLaHora();

    }
}
```

Aquí vemos un ejemplo de clase interna local. La clase "DameLaHora" se encuentra dentro del método "ejecutarTemporizador". Ni siquiera la clase "Reloj" puede acceder a la clase "DameLaHora"

Clases internas anónimas

Como su nombre indica se tata de clases internas sin nombre. Supone rizar el rizo a la hora de simplificar el código de programación y son muy útiles en determinados escenarios.

- **¿Cuándo se utilizan?** Sobre todo al gestionar eventos. Muy utilizadas en aplicaciones gráficas donde el usuario interacciona con controles (botones, menús etc).
- **¿Cuál es su objetivo?** Simplificar el código creando una clase “inline” sobre la marcha. Esto evita tener que crear una clase adicional en nuestro código para después crear una instancia.



```
class Reloj{


    public void ejecutarTemporizador(int intervalo, boolean sonido){


        Timer miTemporizador=new Timer(intervalo, new ActionListener(){

            @Override
            public void actionPerformed(ActionEvent arg0) {
                // TODO Auto-generated method stub
                Date ahora=new Date();

                System.out.println("Te pongo la hora cada 3 segundos: " + ahora);

                if (sonido) Toolkit.getDefaultToolkit().beep();
            }


        });

        miTemporizador.start();


    }
}
```

Clase interna anónima. Tan solo es necesario especificar el nombre de la interfaz que implementa la clase, en este caso "ActionListener"