Clases dentro de clases y métodos. Clases anónimas.

```
public class A {                    //Clase pública. El fichero DEBE llamarse A.java (sólo puede haber una clase pública)

    static class B {                //Clase interna estática
    //TODO Código de la clase B
    }

    class C {                       //Clase interna dinámica. Cada objeto tendrá asociada su propia clase interna
    //TODO Código de la clase C
    }

     A objA = new A();              //Objeto de la clase (externa) A
     B objB = new B();              //Objeto de la clase estática interna B
     C objC = new C();              //Objeto de la clase C (no podría ser estático)
    A anonimo = new A() {           //Objeto de una subclase anónima de A (la referencia "anonimo" lo es "a nivel" A
    //Código de la subclase anónima de A
    };

    void metodo1(E objE) {
       class D {                    //Clase local
       //TODO código de la clase D
       }
         D objD = new D();          //Objeto de la clase local D
         //TODO código del método m (que usará los objetos d y e
    }

    void metodo2() {                //En este método llama a m(.) aportando un objeto de clase anónima como parámetro
        metodo1(new E(){            //el parámetro es un nuevo objeto de una subclase anónima de E
                @Override
                void metodo(){
                //TODO código del método
                };
        });
    }
}

class E {                           //Clase no pública dentro del fichero A.java
     A.B ab = new A.B();            //Objeto de clase interna A.B (no podría ser A.C)
    void metodo(){
    //TODO código del método
    }
}
```

```java
//Clase para mostrar un ejemplo de uso de clases anónimas
//Es un interfaz gráfico (extendemos Frame) con sólo un botón.
//Un objeto de una clase anónima se encarga de atender a la pulsación del botón.

public class EjemploDeClaseAnonima extends java.awt.Frame {

    //El constructor del "frame"
    EjemploDeClaseAnonima() {
        //generamos un botón
        java.awt.Button b = new java.awt.Button("púlsame");
        //y lo añadimos al interfaz gráfico
        add(b);
        //al botón le decimos que añada un objeto que "escuche(atienda) a sus acciones(pulsaciones)"
        b.addActionListener(
                //Generamos un nuevo objeto como si fuera un ActionListener, pero metemos código a continuación
                //Por lo que en realidad estamos generando un objeto de una nueva clase que hereda de ActionListener
                //ActionListener es un Interface, por loq ue se hereda un compromiso concreto que es el método donde
                //ponemos el código que atenderá al botón
                new java.awt.event.ActionListener() {
                    @Override
                    public void actionPerformed(java.awt.event.ActionEvent evt) {
                        System.out.println("Ay!");
                        System.exit(0);
                    }
                }
        );
        //Dos acciones finales para que el interfaz gráfico se organice correctamente(pack) y sea visible
        pack();
        setVisible(true);
    }

    //El main genera un objeto de esta clase de ejemplo
    public static void main(String[] s) {
        new EjemploDeClaseAnonima();
    }
}
```